Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра информатики, информационных технологий и методики обучения информатике

Методика обучения школьников программированию в среде разработки Scratch

Выпускная квалификационная работа

Квалификационная работа допущена к защите
Зав. кафедрой
_____
дата
_____
подпись

Исполнитель: Гурьевских Мария Евгеньевна, обучающийся группы ИНФ-1501Z
_____
подпись

Руководитель: Рожина Ирина Венокентьевна, кандидат педагогических наук, доцент
_____
подпись

Екатеринбург 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном обществе профессия программист востребована и высокооплачиваемая. Но учитывая специфику профессии, важным фактор это саморазвитие, ведь информационные технологии постоянно развиваются, каждый день появляется что-то новое, все большую популярность приобретает программирования. [1 с. 5]

По всему миру десятки миллионов учеников активно участвуют в различных мероприятиях, направленных на получение качественных знаний, которые помогут в будущем получить высокооплачиваемую работу в ИТ-среде. Они ходят на кружки, факультативы по информатике, посещают различные курсы, изучают самостоятельно, смотрят ролики в YouTube с одной целью – получить дополнительные знания в IT сфере. Это модно, престижно, увлекательно, очень развивающее, и это позволит в будущем мире высоких технологий быть всегда в тренде. Даже если ученики в будущей профессиональной деятельности не планируют работать в сфере ИТ, они все равно развиваются и приобщаются к миру технологий, к тому же программирование развивает аналитическое и критическое мышление и память.

Иностранные специалисты приводят множество аргументов, почему детей необходимо обучать программированию. Важным аспектом является использование математического аппарата, то есть визуализация математических навыков, или, например, через выполнение задач программирования приложений Scratch, выработка навыков работы в команде [2 с. 154].

Развитие технологий программирования вызывает необходимость практического изучения не только современных программных средств, но и технологий их разработки. Однако вместе с ростом возможностей сред разработки возрастает их сложность, и как следствие - растет сложность их изучения, что

может приводить к появлению технологического уклона в обучении программирования.

Одним из путей решения проблемы фундаментализации обучения программированию школьников является введение в курс информатики объектно-ориентированного моделирования, с помощью специальных сред программирования с использованием готовых моделей.

Вопросу обучения программирования с помощью специальных учебных сред посвящены работы многих исследователей. В частности, в работах А. Г. Гейна, Н. В. Макаровой, А. П. Шестакова раскрываются методические аспекты обучения программированию с использованием визуальных моделей, в работах А. П. Одинцовой, Т. В. Софроновой, А. А. Тарасовой - геометро- графического моделирования; в работах А. В. Бобровской, Л. В. Кавурко, И. Г. Обойщиковой - математического моделирования в учебных средах программирования; в работах Д. Д. Бычковой, П. В. Кийко, И. В. Паболков, И. А. Теплицкого, Л. Н. Шенгелия показана интегративная роль учебного компьютерного моделирования.

Все это готовит учеников к решению серьезных задач и получение более широких навыков в решении задач, связанных с применением компьютерной техники, предоставляет неограниченные возможности в мир познания информационных технологий. Ведь в Scratch предусмотрена работа с внешним оборудованием, кроме камеры и микрофона - это набор PicoCricket, собранными моделями или отдельными деталями которого можно управлять через Scratch. Это уже серьезная электроника, но в интересной, удобной и познавательной форме.

Объект исследования: процесс развития алгоритмических умений обучающихся на уроках информатики и ИКТ.

Предмет исследования: обучение школьников программированию в среде разработки Scratch.

Цель данной работы: разработка системы занятий для обучения программированию школьников с использованием среды разработки SCRATCH.

Задачи работы:

1. Определить основные достоинства и преимущества использования Scratch в процессе изучения основ и принципов программирования обучающимися общеобразовательной школы на уроках информатики.
2. Проанализировать существующие программы, моделирующие процесс программирования в наглядно-графической форме.
3. Разработать систему занятий для обучения программированию школьников с использованием визуальной среды разработки Scratch;
4. Провести апробацию разработанных материалов.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИСПОЛЬЗОВАНИЯ SCRATCH В ОБУЧЕНИИ

1.1. Предпосылки для использования Scratch в учебном процессе

Стремительное обновление современного образования предполагает публичное и дискуссионное обсуждение предложений по внесению изменений в учебные программы, в частности, в начальных классах.

Анализ обновленной [32] указывает на изъятие и сокращение некоторых тем из программы «Информатика» для общеобразовательных учебных заведений. В пояснительной записке к учебной программе для общеобразовательных учебных заведений 5-9 классов указано, что изучение курса пропедевтики «Информатика» способствует формированию и развитию у школьников ключевых компетентностей (предметная ИКТ-компетентность, межпредметные, коммуникативные и социальные компетентности). Измерение предметной ИКТ-компетентности связывают с технологическими, телекоммуникационными, алгоритмическими и логическими умениями обучающихся. Отметим те, которые относят к алгоритмическим и логическим умениям: формулировка команд для исполнителя, составление алгоритмов по образцу, поиск ошибок в последовательности команд, анализ содержания задач на составление алгоритма для исполнителей, поиск различных вариантов исполнения задач, выбор и обоснование эффективного варианта исполнения; различия алгоритмических структур (следования, циклы, ветвления), создание и выполнение алгоритмов в определенной среде; различения истинных и ложных высказываний.

Перечислим темы по содержанию учебного материала, в процессе изучения которых учащиеся приобретают указанные умения: «Команды и исполнители» (2 класс), «Алгоритмы и исполнители» (3 класс), «Алгоритмы с разветвлением и

повторением» (4 класс), «Основы алгоритмизации» (8 класс), «Алгоритмизация и программирование» (9 класс).

По мнению Е.Д. Патаркина [33], программируя на Scratch перед учениками возникают неограниченные возможности, поскольку кроме подвижных объектов эта среда позволяет создавать целые сказки, с помощью наложения аудио ряда на каждый графический объект. По мере того, как ученики создают программы на языке Scratch, они изучают также основные вычислительные концепции, такие как итерация и условные выражения. Они также получают понимание важных математических понятий, таких как координаты, переменные и стандартные алгоритмические конструкции.

Важно отметить, что ученики изучают эти понятия в содержательном и мотивирующем контексте. Когда учащиеся узнают о переменных в классах традиционной алгебры, они обычно чувствуют небольшую личную связь с концепцией. Но когда они узнают о переменных в контексте Scratch, они могут сразу же использовать переменные очень значимым образом: контролировать скорость анимации или отслеживать счет в игре, которую они создают.

Когда ученики работают над проектами Scratch, они также узнают о процессе проектирования. Как правило, учащийся начинает с идеи, создает рабочий прототип, экспериментирует с ним, отлаживает его, когда что-то идет не так, получает обратную связь от других, затем пересматривает и обновляет его. Это непрерывная спираль: получить идею, создать проект, который приведет к новым идеям, которые приведут к новым проектам, и так далее.

Этот процесс разработки проекта сочетает в себе многие навыки обучения 21-го века, которые будут иметь решающее значение для успеха в развитии ученика в будущем: творческое мышление, четкое общение, систематический анализ, эффективное сотрудничество, итеративное проектирование, непрерывное обучение.

Создание проектов в Scratch также помогает учащимся развить более глубокий уровень владения цифровыми технологиями.

Конечно, большинство обучающихся не станут профессиональными программистами, так же как большинство других учащихся не станут профессиональными писателями, биологами, физиками и химиками. Но обучение программированию дает преимущества для личности ученика: оно позволяет школьникам более полно и творчески выражать свои мысли, помогает им развивать логического мышления и помогает им понять работу новых технологий, с которыми они сталкиваются повсюду в повседневной жизни.

1.2. Анализ программных систем для изучения программирования

С конца двадцатого века большинство ученых сошлось на том, что в обучении программированию школьников среднего и старшего звена следует придерживаться принципов конструктивизма и его разновидности – конструкционизма, что заметно повлияло на педагогический дизайн и воплотилось в следующих педагогических средах:

- Лого и его производные NetLogo и StarLogo;
- Squeak и производный от него Scratch;
- Alice и др.

Сделаем краткий обзор программных систем для изучения программирования.

Название Лого (LOGO) происходит от греческого «логос», что означает «слово», «смысл», «идея».

В литературе термин «Лого» используется в двух значениях:

1) как язык программирования, настолько проста, что ее могут освоить дети, но настолько мощная и выразительная, что и опытный программист найдет в ней много интересного;

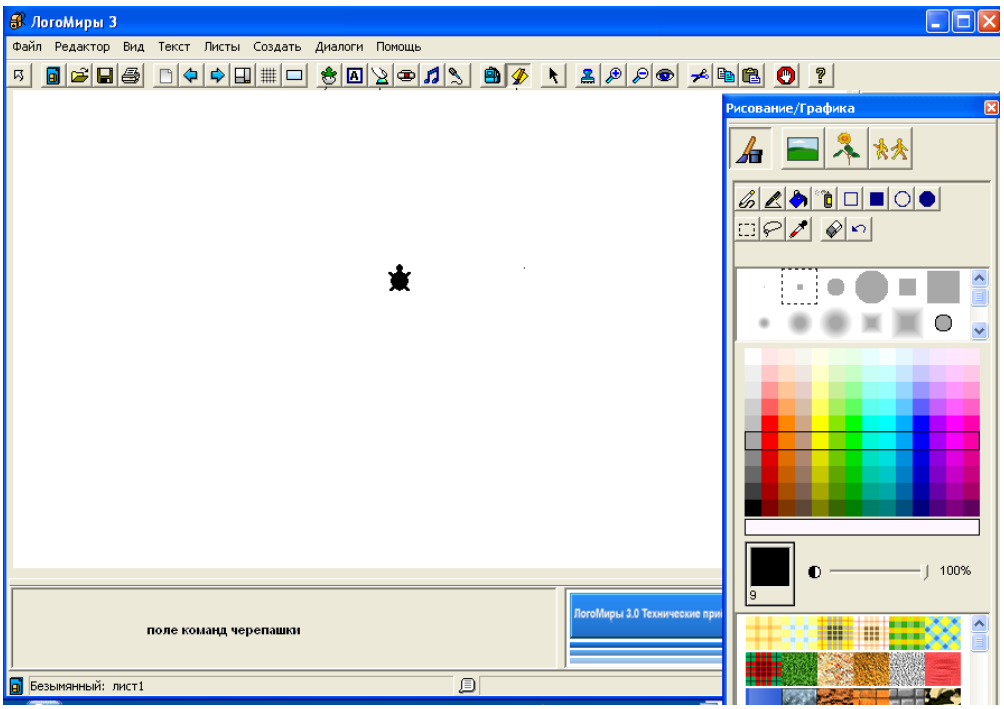
2) как философия обучения, система взглядов на процесс обучения, призванная, как считает С. Пейперт, а не частично улучшить, а коренным образом революционизировать традиционную организацию обучения.

Язык Лого занимает в этой системе центральное место. Философия обучения Лого предполагает преобразование традиционной системы обучения и основывается на идее «использование компьютера как модели, которая может повлиять на наш образ мышления о самих себе» [9].

Объекты среды Лого - это компьютер, «математически говорит за существо» [12] и Черепаха, «кибернетическая животное», управляемая с помощью компьютера.

Обучение в Лого происходит в процессе «бесед» ученика с Черепахой и компьютером в отличие от традиционной организации обучения, в Лого не компьютер управляет процессом обучения, а ученик «учит» компьютер, «сказал ему» на языке Лого.

Черепаха, как и компьютер - объект в среде Лого. (Рисунок 1)



***Р**исунок 1 Пользовательский интерфейс программы ЛОГОМИРЫ 3*

Отправляя команды-сообщения объекту «Черепаха», ученики естественным образом усваивают принципы объектно-ориентированного программирования.

«Работа в микромире Черепахи - это модель изучения идей в тот самый способ, которым мы познаем другого человека. Ученики, которые работают в этой среде, безусловно, открывают в нем интересные факты, приходят к обобщениям, усваивают навыки »[9, с.136].

В начале 1990-х годов М. Резник предложил использовать мультиагентные сообщества ракушек для освоения учащимися экологических стратегий [5]. С множеством ракушек в языке StarLogo ученики могли наблюдать, изучать и моделировать сложные физические, химические, биологические и социальные феномены. Хотя речь создавалась в первую очередь как средство обучения, в этой среде оказалось возможным ставить и серьезные эксперименты даже в области мультагентного моделирования. Но это скорее относится к возможностям среды, а не к практическим реализациям большинства школьников, которые с удовольствием заставляли двигаться черепашек и ракушек и часть из них сегодня довольно уверенно чувствует себя в среде ИТ-специалистов.

Исследовательские возможности StarLogo получили дальнейшее развитие в NetLogo, в котором в последние годы были построены различные исследовательские модели, которые использовались в научных статьях и обсуждались в книгах по мультиагентного моделирования и социологии [6].

Библиотека моделей, созданных в среде NetLogo, большая и пополняется не только разработчиками, но и членами сообщества (логосфера), в котором можно: прочитать описание модели, ее назначения, положенные в основу принципы, посмотреть выполнения программы в сети; скачать модель и запустить на своем компьютере внести в модели изменения и использовать готовые процедуры, взятые из чужой модели, для своих собственных нужд; загрузить свою модель на сервер и предложить ее на обсуждение и совместного использования. Однако

среду NetLogo следует рассматривать как игрушку для взрослых, в которой довольно успешно обучались и развивались студенты старших курсов технических вузов, в отличие от школьников, для которых данная среда достаточно сложная.

В 1968 году А. Кей, вдохновленные идеями С. Пейперта, придумал Dynabook - «персональный компьютер для детей любого возраста» [2].

А. Кей видел роль персонального компьютера как личностного динамической среды (метамедиа), что объединяло в себе все другие среды: текст, графику, анимацию и даже то, что еще не изобретено [3].

Так же, как и Лого, речь Smalltalk, разработанная как программная часть проекта Dynabook, является одновременно и языком программирования, и средой разработки программ. Это чисто объектно-ориентрованная среда, в которой абсолютно все рассматривается как объекты.

Как отмечает один из ее разработчиков Д. Ингаллс, «цель проекта Smalltalk - сделать мир информации доступным для детей любого возраста.

Все трудность состоит в том, чтобы найти и применить достаточно простые и эффективные метафоры, которые позволят человеку свободно оперировать самой разнообразной информации от чисел и текстов к звуковым и зрительным образам»[4].

В основу языка положены две простые идеи:

1) все являются объектами;

2) объекты взаимодействуют, обмениваясь сообщениями.

Современной реализацией Smalltalk является Squeak [11], в котором появляется все больше свойств проекта Dynabook - мощная 2D- и 3D-графика, многоголосый и синтезированный звук, поддержка анимации и видео, средства для работы с различными медиа-форматами и т.д. [8] .

Scratch - еще одна среда программирования, созданная под руководством М. Резника [7]. Scratch позволяет детям создавать собственные анимированные и интерактивные истории, игры и другие творения. Основная задача проекта - стать частью образовательной программы для детей и подростков, развить в них творческие способности, логическое мышление и свободу в использовании информационных технологий. Все это предлагается развить путем привлечения учащихся к процессу конструирования интерактивных презентаций, мультиков, игр. Дети могут составлять свои программы из блоков команд («кирпичиков») так же, как они строили домики и машинки из деталей «Лего».

По своей внутренней архитектуре Scratch базируется на Squeak (Рисунок 2).

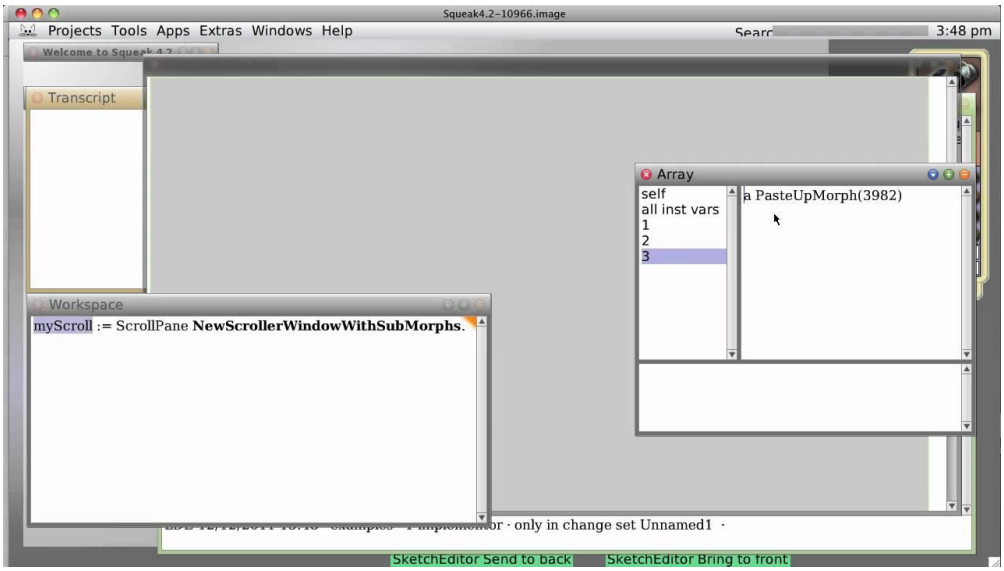


Рисунок 2 Пользовательский интерфейс программы Squeak 4.2

Squeak — язык программирования, диалект языка Smalltalk. Имеет кросс-платформенную реализацию (Windows, Linux, Macintosh).

Первоначально Squeak был разработан группой программистов Apple Computer, в которую входили некоторые разработчики Smalltalk-80. Разработка была продолжена той же группой уже в Walt Disney Imagineering.

На данный момент Squeak доступна абсолютно бесплатно для любого использования. Кроме того, Squeak полностью доступен в исходных кодах (в том

числе и виртуальная машина). В Squeak реализовано несколько графических подсистем (в том числе MVC, унаследованная от оригинального Smalltalk-80, в текущей версии не поддерживается, работает в версиях младше 3.8). Однако основной является собственная графическая подсистема Morphic (портированная из Self).[34]

Поэтому при «исчерпании» возможностей Scratch по мере развития навыков программирования можно перейти к родительской мультимедийной среде объектно-ориентированного моделирования Squeak, используя мощные средства ООП языка Smalltalk. Так же, как и в NetLogo, вокруг Scratch существует сообщество.

Scratch приучает собирать проект из кирпичиков и делиться результатами своих действий с другими людьми. Эти навыки важны не только внутри специальных сред программирования, но и в современных сетевых сообществах. Единство процессов создания, поиска и хранения информационных кирпичиков все чаще можно наблюдать на страницах современных сайтов, использующих концепцию Web 2.0. Метафора строительных блоков, из которых дети и взрослые могут собрать простые и очень сложные конструкции, присутствует не только в учебных проектах, но и в большинстве современных сетевых сервисов форматов Web 2.0, предназначенных для поддержки организаций и сетевых сообществ обмена знаниями:

1) обучающиеся представляют и представляют, что именно они хотят сделать и получить в результате;
2) обучающиеся создают проект, основанный на своих представлениях;
3) обучающиеся играют с результатами своей деятельности;
4) обучающиеся делятся результатами своей деятельности с другими людьми;
5) обучающиеся обдумывают и обсуждают свои результаты;

6) обсуждения и обдумывания приведет к новым представлениям и новым проектам.

Среда разработки Alice (Рисунок 3), так же, как и Scratch - относительно новый проект, разрабатываемый в университете Карнеги-Меллона. В отличие от своих предшественников, Alice - полностью трехмерная среда моделирования. Alice 2.2 позиционируется разработчиками как средство обучения объектно-ориентрованного программирования, а Alice 3 - как средство объектно-ориентрованного моделирования [10]. Alice является средой, в которой можно манипулировать 3D объектами (двигать, вращать, менять цвет и т.д.) и создавать программы, которые генерируют анимацию в виртуальных мирах.

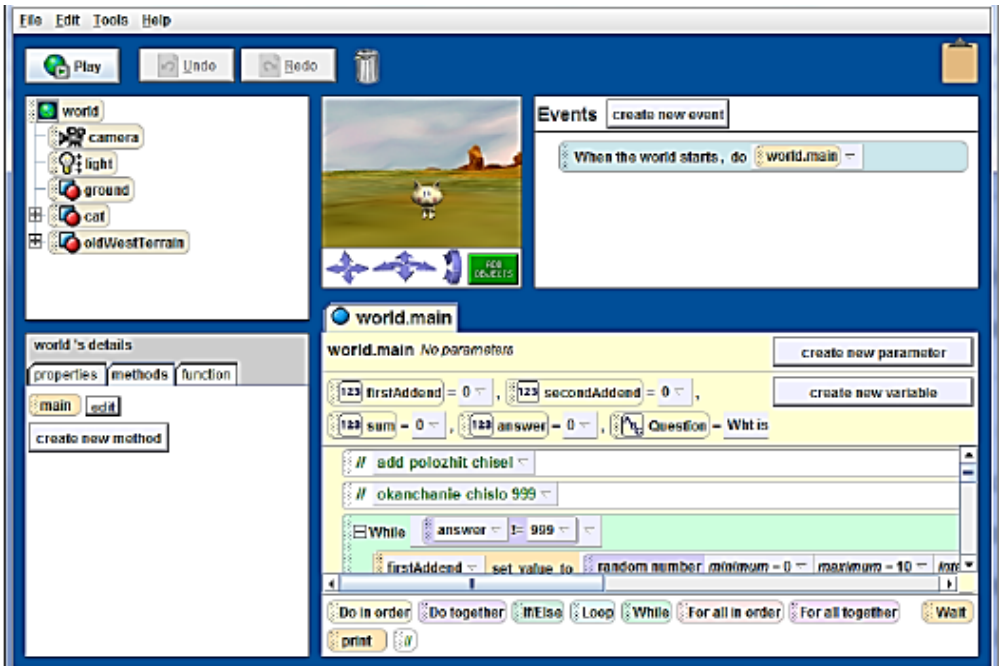


Рисунок 3 Пользовательский интерфейс программы Alice 2.2

Рабочая плоскость Alice разбита на несколько окон: в одном в реальном времени отображается виртуальный мир; в другом присутствует дерево объектов, и для каждого из них предоставляется набор доступных методов, функций и свойств; центральная часть отведена под редактор исходного кода.

Среда программирования Alice выпускается для Windows, Mac OS и Linux в двух редакциях: основной (для вузов) и упрощенной (для школ). В среде имеется большая библиотека объемных объектов из реального мира (природа, животные, инструменты быта и т.д.). Их можно двигать, вращать, менять цвет и размер при помощи мыши, а на основе полученного виртуального мира программно описывать анимацию и создавать игровые модели.

Окна Alice очень похожи на те, что встречаются в профессиональных современных интегрированных средах разработки визуальных приложений типа Visual Studio.

Созданный проект запускается на исполнение в отдельном окне при помощи кнопки Play. В среде Alice используется собственный встроенный язык программирования, приближенный к синтаксису языков современных объектно-ориентированных языков программирования таких, как Java, C++ или Visual Basic. Поскольку программное обеспечение Alice позволяет создавать только синтаксически правильные команды, то программирование в данной среде сводится лишь к разработке и реализации соответствующих алгоритмов. [4]

Учащемуся не нужно запоминать синтаксис какой-либо конструкции, он всегда может воспользоваться набором имеющихся процедур и функций рассматриваемого объекта или всплывающими подсказками. Именно это позволяет школьникам в дальнейшем сконцентрировать свое внимание на сценарии игры, сцены, используемых объектах, их свойствах и методах, а не беспокоиться о синтаксических ошибках своих приложений.

Программирование в Alice отчасти напоминает работу в среде Scratch. Процесс написания игры напоминает игру «Пазлы», когда необходимо состыковать между собой программные элементы, из которых собирается вся программа

От учащегося лишь иногда требуется ввести с клавиатуры константы или название собственных идентификаторов. Для наглядности представленного кода блоки программных элементов выделяются определенным цветом в зависимости от их типа. Эти блоки можно легко перетаскивать, меняя порядок и вложенность. [13]

Таким образом, Alice – игровая среда программирования, создающая игры и анимационные фильмы, обучающая основам объектно-ориентированного программирования, позволяет передать поведение реального мира, описывая взаимодействия объектов, скрывая детали реализации, разрабатывать программное обеспечение повышенной сложности за счет улучшения его технологичности (лучших механизмов разделения данных, увеличения повторяемости кодов, использования стандартизованных интерфейсов пользователя и т.д.).

Alice как конструктор игр, редактор написания собственных историй является инструментом, посредством которого можно повысить уровень алгоритмического мышления, развить коммуникативные умения при создании коллективных проектов (выслушать товарища, доказать свою точку зрения, оценить проект, правильность выполнения задания и пр.), развить личностные качества (умение ставить и формулировать новые задачи, интересы своей познавательной и регулятивной деятельности; умение самостоятельно планировать пути достижения определенных целей в обучении, находить эффективный способ решения задачи; умение критично относиться к результатам своей деятельности в процессе достижения результата).

1.3. Использование Scratch во внеурочной деятельности и на уроках информатики

Одной из учебных сред, используемых на уроках информатики является «Scratch» - объектно-ориентированное визуальная среда программирования.

Можно начинать пользоваться языком с нуля, не обладая никакими предварительными знаниями в программировании. Как уже писалось ранее, в среде Scratch используется метафора кирпичиков Лего, из которых даже молодые дети могут собрать простые конструкции. Но, начав с малого, можно дальше развивать и расширять свое умение строить алгоритмы и программировать [3 с. 244]. Scratch создавался специально для того, чтобы подростки 10 - 16 лет использовали его самостоятельно в сети внешкольного обучения. Важно отметить, что Scratch приходит в современный мир вместе с другими важными педагогическими новациями. Согласно идеологии этого движения, ребенок должен осваивать не программы, а различные способы деятельности: создавать свои собственные истории, придумывать игры, разрабатывать компьютерные и алгоритмические модели. Данная система проста для восприятия даже детям в младшей школе, ведь все операторы языка и другие его элементы представлены блоками, которые могут соединяться друг с другом, образуя скрипт (фрагмент кода) [4 с. 422]. Программируя на Scratch ученики получают понятия алгоритмизации и программирования, создают игры, анимации или музыку.

Выбор программной среды обуславливается простым интерфейсом, доступностью и скоростью создания подвижных объектов, а также легкостью установления и применения программы. К тому же, представленное программное обеспечение является бесплатным.

Технологии анимации позволяют реализовать принцип наглядности. Их эффективность заключается в том, чтобы сосредоточить внимание ученика на

определенном объекте, заинтересовать его, стимулировать активность, способствуют быстрому запоминанию материала.

Определим требования к применению и созданию анимации в программе Scratch:

- доступность, простота применения;
- соблюдение методических правил по количеству анимационных изображений, времени, выделенного на их применение;
- соответствие подобранных анимационных изображений теме и дидактическим целям и задачам занятия;
- соответствие анимационных изображений возрастным особенностям детей;
- качество анимационных изображений, обоснованность и рациональность применения [5 с. 381].

Более подробно возможности программирования в Scratch освещают в своих работах такие ученые, как Е. Д. Патаракина, Т. Е. Сорокина [10, 11, 12].

В литературе представлены различные направления работы в среде Scratch:

- разработка интерактивных дидактических материалов,
- реализация графики,
- моделирования,
- проектирования,
- построение социальной сети для обмена собственными проектами,
- поддержка самостоятельного обучения учащихся,
- работа в сотрудничестве
- другие [4, 5, 6, 9, 13] .

Анализ публикаций в иностранных изданиях свидетельствует о накоплении значительного опыта внедрения среды программирования Scratch в учебный

процесс, а именно по этой тематике разработаны курсы в Гарвардском университете, Калифорнийском университете в Беркли, колледже Нью-Джерси других [1, 2, 3].

Использование среды Scratch в начальных и средних классах, а также изучение элементов программирования со старшими школьниками на факультативных занятиях вызывает необходимость показать возможности работы в среде программирования, сочетая изучение алгоритмизации (алгоритмических структур: ветвления, циклы другие) и элементов графического интерфейса.

Описание Scratch

Scratch является средой объектно-ориентированного визуального программирования, которое предоставляет возможности создавать компьютерные анимации, мультимедийные презентации, интерактивные материалы в виде историй и игр, модели и др.

Scratch является свободно распространяемой в учебных целях программой, которую можно скачать с официального сайта разработчиков по адресу: http://scratch.mit.edu (https://scratch.mit.edu/scratch_1.4/).

На сайте есть возможность создавать проект в он-лайн режиме. Отметим, что в программе предусмотрен выбор языка, поэтому программирование осуществляется на понятном пользователю языке.

Программирование в указанном среде происходит следующим образом: пользователи «собирают» программу (скрипт) из блоков, также некоторые блоки можно встраивать друг в друга, исключая возможность возникновения синтаксических ошибок. В среде Scratch исполнителями являются спрайт(-ы) и сцена.

Объект, который связывают с определенным изображением и набором переменных и скриптов для определения его поведения называют спрайтом. По умолчанию таким исполнителем выступает так называемый «рыжий кот»,

который может двигаться, говорить, изменять внешний вид и взаимодействовать с другими исполнителями на сцене.

Спрайт можно менять, импортировав его образ из встроенной библиотеки, в которой предусмотрены следующие категории: животные, фантастика, буквы, люди, вещи, транспорт. Также содержание библиотеки можно дополнять другими файлами. Обращаем внимание, что образ спрайта можно создавать, используя встроенный графический редактор или другие программы.

Команды, которые выглядят как блоки, объединенные в определенные группы:

- «Движение» (осуществление перемещения спрайтов),
- «Вид», (смена образов спрайта, его текстовые диалоги)
- «Звук» (звуковые команды, громкость, темп),
- «Карандаш» (построение графических изображений),
- «Управление» (организация циклов, ветвлений),
- «Датчики» (информация о соприкосновения объектов и определения расстояния между ними),
- «Операторы» (осуществление математических и логических операций, выбор случайного числа),
- «Переменные» (создание переменных, назначение определенных значений).

Выводы по главе 1

В 1 главе были рассмотрены основные теоретические основы использования Scratch в обучении. Нами были проанализированы следующие педагогические среды:

- *Лого и его производные NetLogo и StarLogo.*

Минус этой среды заключаются в том, что для школьников, данная среда достаточно сложная. NetLogo следует рассматривать как игрушку для студентов или учеников старших классов.

- *Alice.*

Alice является средой, в которой можно манипулировать 3D объектами (двигать, вращать, менять цвет и т.д.) и создавать программы, которые генерируют анимацию в виртуальных мирах. Данная среда имеет широкое распространение в странах США.

- *Squeak и производный от него Scratch.*

Scratch является свободно распространяемой в учебных целях программой.

Огромным плюсом в изучении данного языка является его простота. Scratch является подходящей средой программирования как для начальных, так и для средних классов.

Scratch можно рассматривать как инструмент для создания проектов. Данная среда обладает большим количеством возможностей. Можно создавать: игры, презентации, мультфильмы, открытки.

Рассмотренный в первой главе теоретический минимум служит основной составляющей разработки пропедевтического курса по изучению Scratch в 8 классе.

ГЛАВА 2. ПРАКТИЧЕСКИЕ ВОПРОСЫ ИСПОЛЬЗОВАНИЯ SCRATCH ПРИ ИЗУЧЕНИИ ПРОГРАММИРОВАНИЯ В 8 КЛАССЕ

2.1. Программа курса по изучению Scratch в 8 классе

Среда Scratch является кросс-платформенным программным обеспечением, поддерживающим рускоязычный интерфейс и распространяется бесплатно.

Выбор программной среды остается на усмотрение учителя с учетом имеющейся компьютерной техники и уместной для конкретного учебного заведения или класса траектории обучения.

При преподавании темы «Алгоритмы и их исполнители», кроме учебника, целесообразно использовать учебные пособия для использования в общеобразовательных учебных заведениях, в которых более подробно предоставлены сведения по теме «Алгоритмизация и программирование», а также освещены особенности учебной среды Scratch.

Программа по информатике 8 класс (на основе авторской программы Босовой Л.Л., Босова А.Ю. Информатика. Программа для основной школы: 5-6 классы. 7-9 классы. – М.: БИНОМ. Лаборатория знаний, 2013)

ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ С ОПРЕДЕЛЕНИЕМ ОСНОВНЫХ ВИДОВ УЧЕБНОЙ ДЕЯТЕЛЬНОСТИ
8 класс

№ п/п	Название раздела, темы	Кол-во часов	Основные виды деятельности
1.	Математические основы информатики	13	*Аналитическая деятельность:* • анализировать любую позиционную систему как знаковую систему; • определять диапазон целых чисел в n-разрядном представлении; • анализировать логическую структуру высказываний; • анализировать простейшие электронные

			схемы. *Практическая деятельность:* • переводить небольшие (от 0 до 1024) целые числа из десятичной системы счисления в двоичную, восьмеричную, шестнадцатеричную и обратно; • выполнять операции сложения и умножения над небольшими двоичными числами; • строить таблицы истинности для логических выражений; • вычислять истинностное значение логического выражения.
2.	Основы алгоритмизации	10	*Аналитическая деятельность:* • приводить примеры формальных и неформальных исполнителей; • придумывать задачи по управлению учебными исполнителями; • выделять примеры ситуаций, которые могут быть описаны с помощью линейных алгоритмов, алгоритмов с ветвлениями и циклами; • определять по блок-схеме, для решения какой задачи предназначен данный алгоритм; • анализировать изменение значений величин при пошаговом выполнении алгоритма; • определять по выбранному методу решения задачи, какие алгоритмические конструкции могут войти в алгоритм; • осуществлять разбиение исходной задачи на подзадачи; • сравнивать различные алгоритмы решения одной задачи. *Практическая деятельность:* • исполнять готовые алгоритмы для конкретных исходных данных; • преобразовывать запись алгоритма с одной формы в другую; • строить цепочки команд, дающих нужный результат при конкретных исходных данных для исполнителя арифметических действий; • строить цепочки команд, дающих нужный результат при конкретных исходных данных для исполнителя, преобразующего строки символов; • составлять линейные алгоритмы по управлению учебным исполнителем; • составлять алгоритмы с ветвлениями по управлению учебным исполнителем; • составлять циклические алгоритмы по

			управлению учебным исполнителем; • строить арифметические, строковые, логические выражения и вычислять их значения; • строить алгоритм (различные алгоритмы) решения задачи с использованием основных алгоритмических конструкций и подпрограмм.
3.	Начала программирования	10	*Аналитическая деятельность:* • анализировать готовые программы; • определять по программе, для решения какой задачи она предназначена; • выделять этапы решения задачи на компьютере. *Практическая деятельность:* • программировать линейные алгоритмы, предполагающие вычисление арифметических, строковых и логических выражений; • разрабатывать программы, содержащие оператор/операторы ветвления (решение линейного неравенства, решение квадратного уравнения и пр.), в том числе с использованием логических операций; • разрабатывать программы, содержащие оператор (операторы) цикла; • разрабатывать программы, содержащие подпрограмму; • разрабатывать программы для обработки одномерного массива: • нахождение минимального (максимального) значения в данном массиве; • подсчёт количества элементов массива, удовлетворяющих некоторому условию; • нахождение суммы всех элементов массива; • нахождение количества и суммы всех четных элементов в массиве; • сортировка элементов массива и пр.

Программа изучения основ программирования с использованием Scratch

№	Тема занятия	Форма	Часы
1.	Интерфейс Scratch	Лекция	2
2.	Знакомство с анимацией	Лекция, комбинированный	2
3.	Путешествие в космос	Практика	2
4.	Рисование собственного спрайта	Практика	2
5.	Проект «Часы»	Практика	2

2.2. Примеры уроков по изучению Scratch

Урок лекция 1. Интерфейс Scratch

Тема: создание собственного объекта в среде программирования Scratch.

Цель: научиться создавать в среде программирования Scratch собственные объекты и анимировать их.

Оборудование: ПК с установленными ОС и Scratch, (данная) инструкция.

Структура урока:

1. Организационный момент.
2. Актуализация опорных знаний.
3. Изучение нового материала.
4. Инструктаж по ТБ.
5. Закрепление изученного материала.
6. Подведение итогов урока.
7. Домашнее задание.

Ход урока

1. Организационный момент

Приветствие класса. Проверка присутствия и готовности учащихся к уроку. Проверка выполнения домашнего задания.

2. Актуализация опорных знаний

Спрайт - это объект Scratch, связанный с изображением, набором переменных и скриптов, которые определяют его поведение.

Как устойчиво используют специального исполнителя указаний - **Рыжего кота**  . Он может двигаться, говорить, изменять внешний вид, взаимодействовать с другими исполнителями на сцене. Других исполнителей можно привлекать:

• из библиотеки Scratch;

• из сети (локальной или глобальной)

• как объекты, созданные в других графических программах (например, GIMP)

• как объекты, созданные в графич редакторе, встроенном в Scratch.

Скрипт (script, сценарий, метод) - последовательность указаний, определяющий, какие действия и в каком порядке нужно выполнить к определенному объекту (спрайта).

Скрипты создают методом сообщением отдельных блоков: либо последовательно, либо располагая блок в определенном месте другого блока (структуры, управляющей функции и т.д.). Один спрайт может иметь несколько скриптов, которые запускают независимо действием пользователя (нажатием клавиши или кнопки мыши), таймером или получением сообщения от другого спрайта. Скрипт состоит из стеков.

Стек (stack) — набор последовательно соединенных разноцветных графических блоков в пределах одного события.

Блок - это минимальный фрагмент программы в Scratch: переменная, оператор, функция или структура, управляющая.

Событие - нажатие на заданную клавишу (например, пробел), щелчок по исполнителю или по сцене, поступления сообщения от другого исполнителя и т.д.

Внешний вид исполнителей можно менять, используя разные образы (на английском costume). Образы, как и исполнителей, можно выбирать из галереи, нарисовать или загрузить.

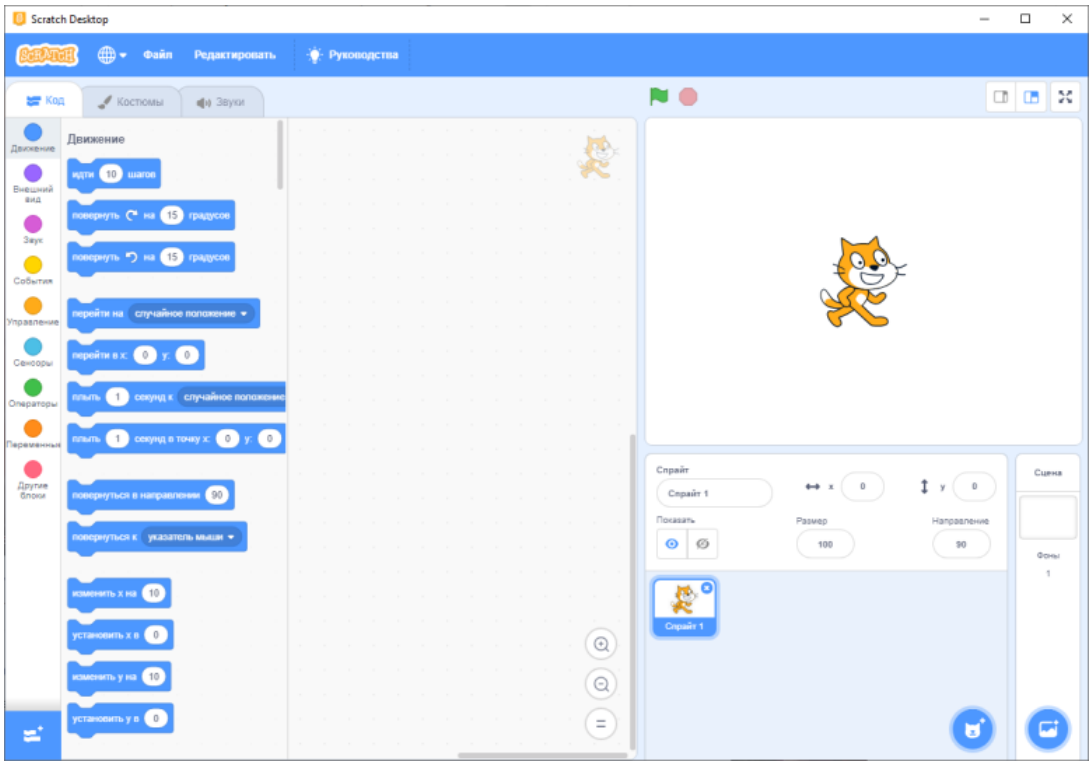
Образы (вид спрайта) - совокупность изображений одного и того же объекта (спрайта), каждое из которых несколько отличается от предыдущих.

Звуки - присоединены звуковые эффекты и музыка.

Сцена - область, в которой действует объект (спрайт) при выполнении программы.

3. Изучение нового материала

При запуске среды программирования Scratch открывается программное окно с новым проектом, который содержит одного исполнителя.



На программном окне расположены следующие кнопки:

 — установить язык общения с пользователем;

 — сохранить этот проект;

 — поделиться этим проектом;

 — дублировать объект;

 — удалить объект;

 — увеличить объект;

 — уменьшить объект;

 — уменьшить масштаб отображения;

 — увеличить масштаб отображения;

 — перейти к просмотру пректа в полноэкранном режиме;

 — запуск скрипта;

 — остановка скрипта;

 — можно повернуться;

 — поворачиваться только направо/налево;

 — не поворачиваться;

 — запретить;

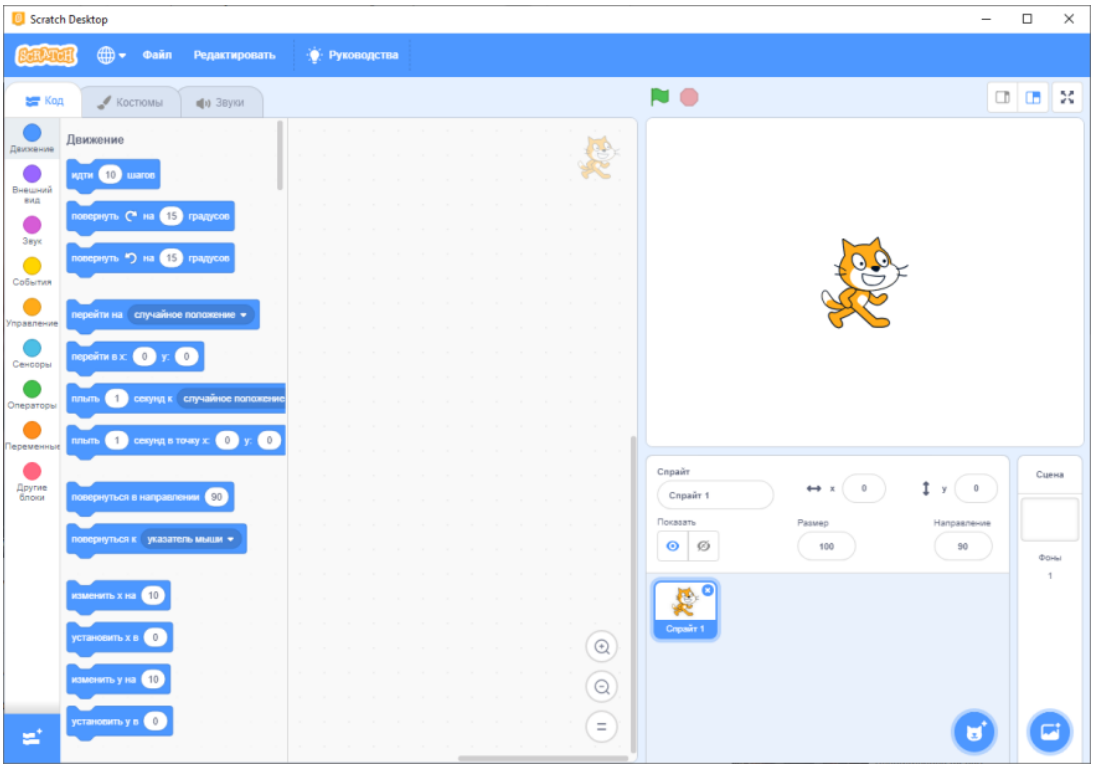
 — нарисовать новый объект;

 — выбрать новый объект из файла;

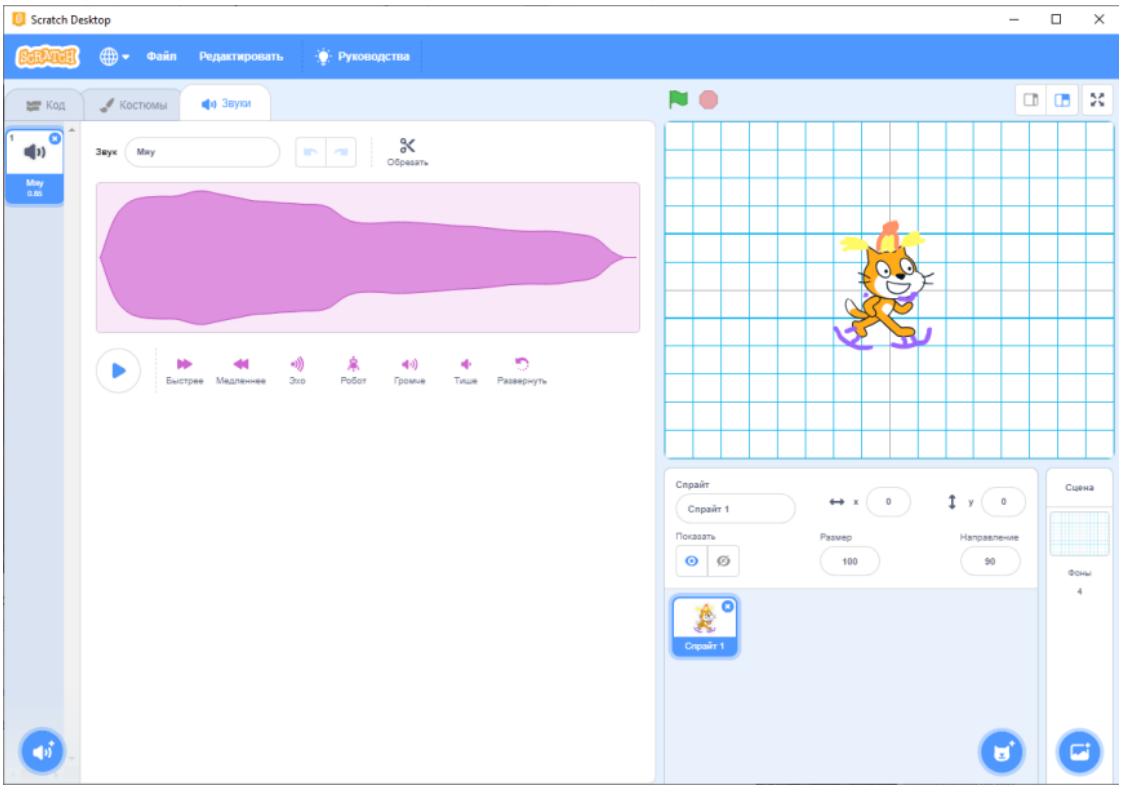
 — установить новый объект.

Окно настройки объекта содержит такие вкладки:

Скрипты - область создания скриптов;



Звуки - область, в которой можно добавлять звуки к объекту

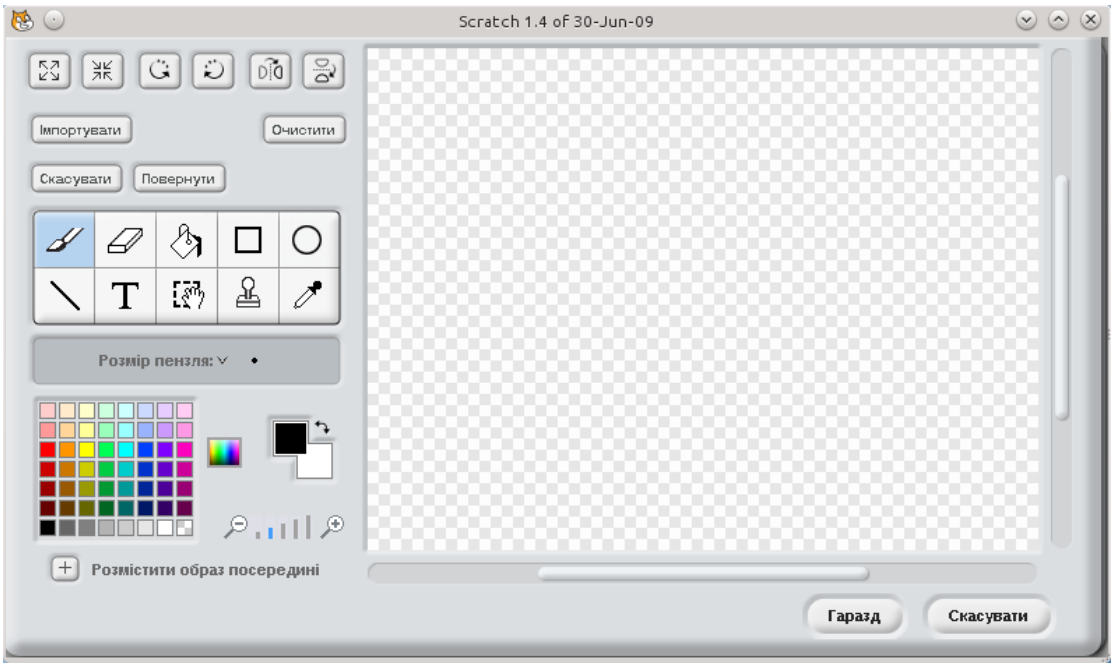


.В правой верхней части рабочего окна находится Сцена. Именно на ней происходят все события. На ней располагают объекты, для каждого проекта подбирают декорации (фон).

В правой нижней части рабочего окна можно увидеть все объекты текущего проекта, их названия.

Объектами и фону можно задать указания. Указания блоки разделены на категории (группы), кнопки вызова которых расположены в верхней левой части окна. Если нажать кнопку, то в левой нижней части окна можно увидеть все указания-блоки данной категории. Эту часть называют контейнером блоков. Для того, чтобы собрать из отдельных блоков сценарий (скрипт), нужно перетащить блоки в центральную часть рабочего окна на закладку Скрипты.

Для создания (рисования) объекта собственноручно нужно вызвать встроенный графический редактор, нажав кнопку  - Нарисовать новый объект.



На программном окне этого графического редактора (см. Рисунок выше) расположены такие кнопки:

	увеличение размера;
	уменьшение изображения
	повернуть изображение против часовой стрелки;
	повернуть изображение по часовой стрелке;
	зеркальное отражение (вертикальное зеркало)
	зеркальное отражение (горизонтальное зеркало)
	кисть (голубым цветом обозначены активное состояние)
	ластик;
	заливка;

	прямоугольник
	эллипс;
	линия;
	текст;
	выделить область;
	штамп;
	пипетка;
	палитра цветов;
	увеличить или уменьшить масштаб просмотра рисунка
	указатель цвета
	задать для исполнителя точку, будет центром вращения при выполнении указаний.

Scratch- это визуальный язык программирования для детей которую изобрели в MIT, абсолютно уникальная среда для обучения программированию. Отличие ее от других систем программирования заключается в том, что она полностью визуальная. Изучение программирования с ней превратится в увлекательное занятие или игру для детей.

Урок лекция 2. Знакомство с анимацией

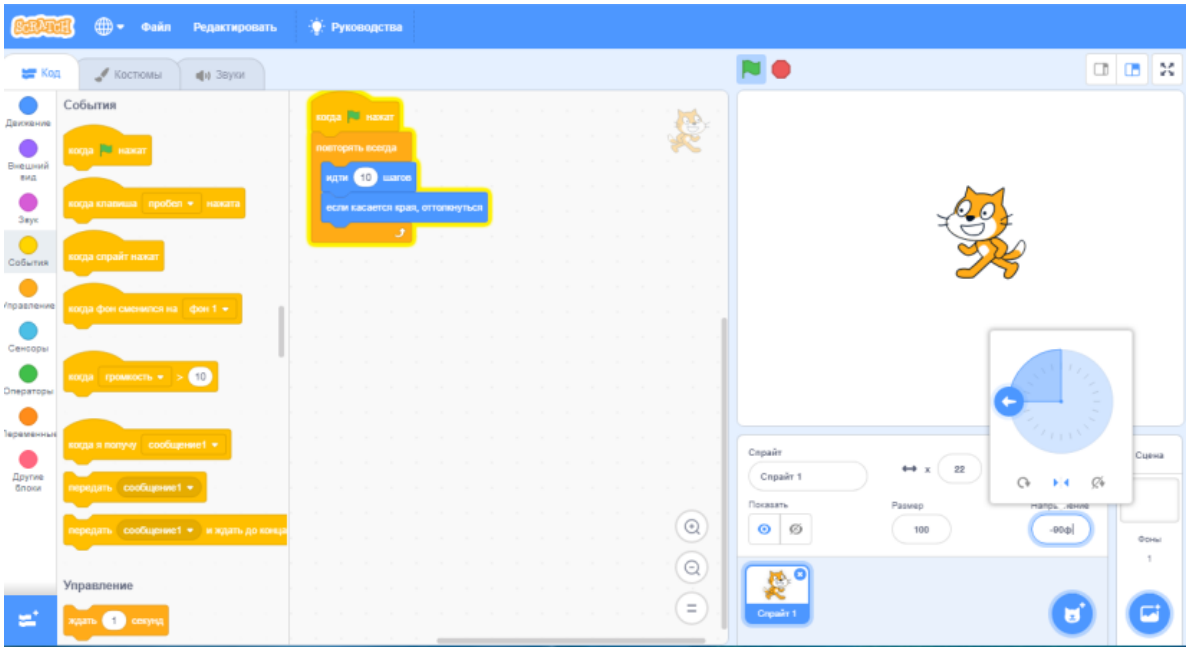
Мы хотим, чтобы объект двигался всегда, пока мы не нажмем кнопку остановки (красный круг). Для этого надо поступить так: вытащить команду *иди 10 шагов* из цикла *повтори ...*, убрать цикл *повтори ...* (перетащить его в левый столбец), взять команду *всегда* и собрать такую программу:

Теперь программа должна работать вечно, пока не будет остановлена. Так и происходит. Если нажать кнопку запуска, то зеленый флажок будет гореть до тех пор, пока не будет нажат красный.

С кнопкой **движение** связана такая команда как *если край, оттолкнуться*.

Если ее поместить в цикл *всегда*, то объект уже не остановится достигнув края, а оттолкнется от него и пойдет в другую сторону. Составьте вот такой скрипт и посмотрите, как он работает.

Скрипт работает замечательно, но кот не совсем нормален при движении справа налево. Он идет вниз головой. Для исправления этого недочета остановим программу и снова обратим свой взгляд на ячейку свойств объекта.



В левой части этой ячейки есть три маленькие кнопки: со скругленной стрелкой, двуголовой стрелкой и квадратной точкой. По умолчанию нажата первая кнопка и поэтому объект при столкновении

поворачивается так, как мы наблюдали. Если нажать вторую кнопку, то он будет поворачиваться так, как нам надо в данный момент (слева направо). Третья кнопка вообще запрещает какие-либо повороты. Нажмите вторую кнопку и посмотрите, как ходит кот. Не забудьте после этого остановить программу.

Координаты

Когда окно Scratch только открылось, мы видим, что кот стоит в центре холста. Посмотрим теперь на его координаты *x* и *y* в ячейке свойств объекта. Там написано, что **x: 0 и y: 0**. В Scratch начало системы координат - это центр холста. От центра вправо значение *x* увеличивается, влево — уменьшается (становится минусовым, отрицательным). Аналогично с осью *y*. По направлению вверх идут положительные значения, вниз — отрицательные. Чем дальше от центра, тем больше абсолютное значение числа. Переместите несколько раз кота по холсту и отметьте, как меняются значения *x* и *y* в ячейке свойств объекта.

Сцена

Чтобы было легче ориентироваться в координатах, проделаем кое-какие манипуляции с холстом - наложим на него картинку с системой координат.

Эта нижняя правая ячейка окна Scratch предназначена для работы с объектами текущего проекта. Здесь можно добавлять новые объекты и переключаться между теми, которые у нас уже имеются. У нас пока существуют только кот и сцена. Переключаться между ними можно, щелкая по ним левой кнопкой мыши.

Новые объекты.

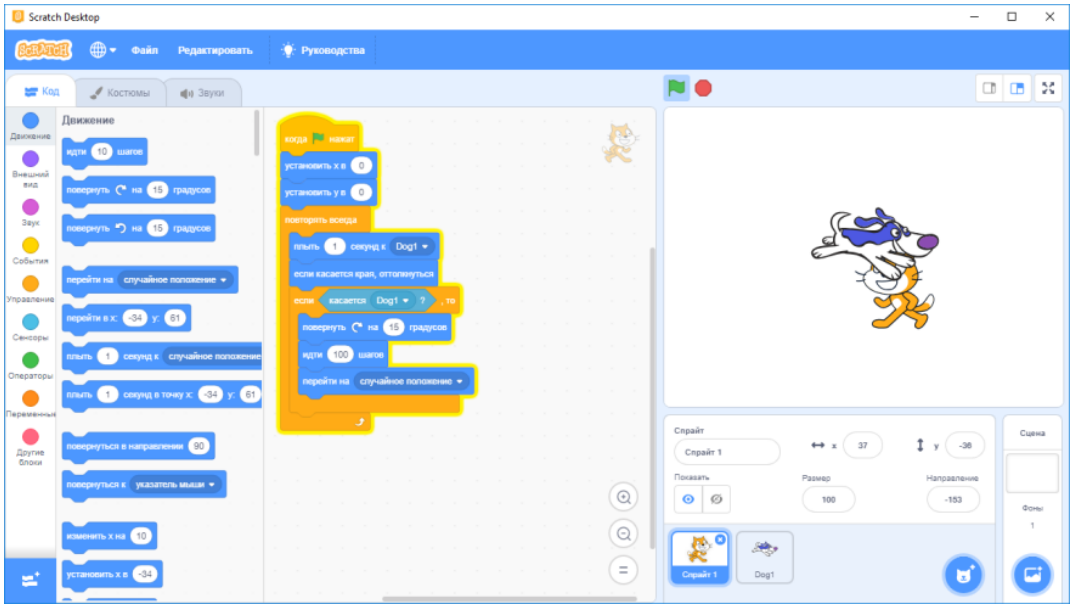
Теперь предположим, что на сцене у нас будет бегать помимо кота еще какая-нибудь живность.

Как в Scratch добавить новые объекты? Следует нажать на вторую кнопку в ряде кнопок под холстом

После чего перед нами открывается диалоговое окно, где из папок *Animals* (животные), *Fantasy* (фантазия), *Letters* (буквы), *People* (люди), *Things* (вещи) и *Transportation* (транспорт) можно выбрать любой объект. Давайте пока ограничимся только папкой *Animals* и добавим на холст какое-нибудь животное, птицу или насекомое. Сделайте это.

Объект добавляется в центре холста. Теперь дайте ему соответствующее имя в ячейке свойств объекта.

Добавим собаку. Хорошо бы, чтобы второй объект - собака тоже как-нибудь двигался. Поскольку мы уже запрограммировали кота, то можно не составлять новый скрипт, а скопировать программу кота и перенести копию на новый объект. Делается это так. Переключаемся на кота → щелкаем правой кнопкой по скрипту и в контекстном меню выбираем команду **дублировать** → перемещаем курсор мыши с прилипшей к нему копией на иконку второго объекта в нижней правой ячейке (будьте внимательны: при этом вокруг иконки должна появиться серая рамка!) и щелкаем мышью.



Вопросы к самостоятельной работе:

1.Сколько циклов в программе? Назовите их.

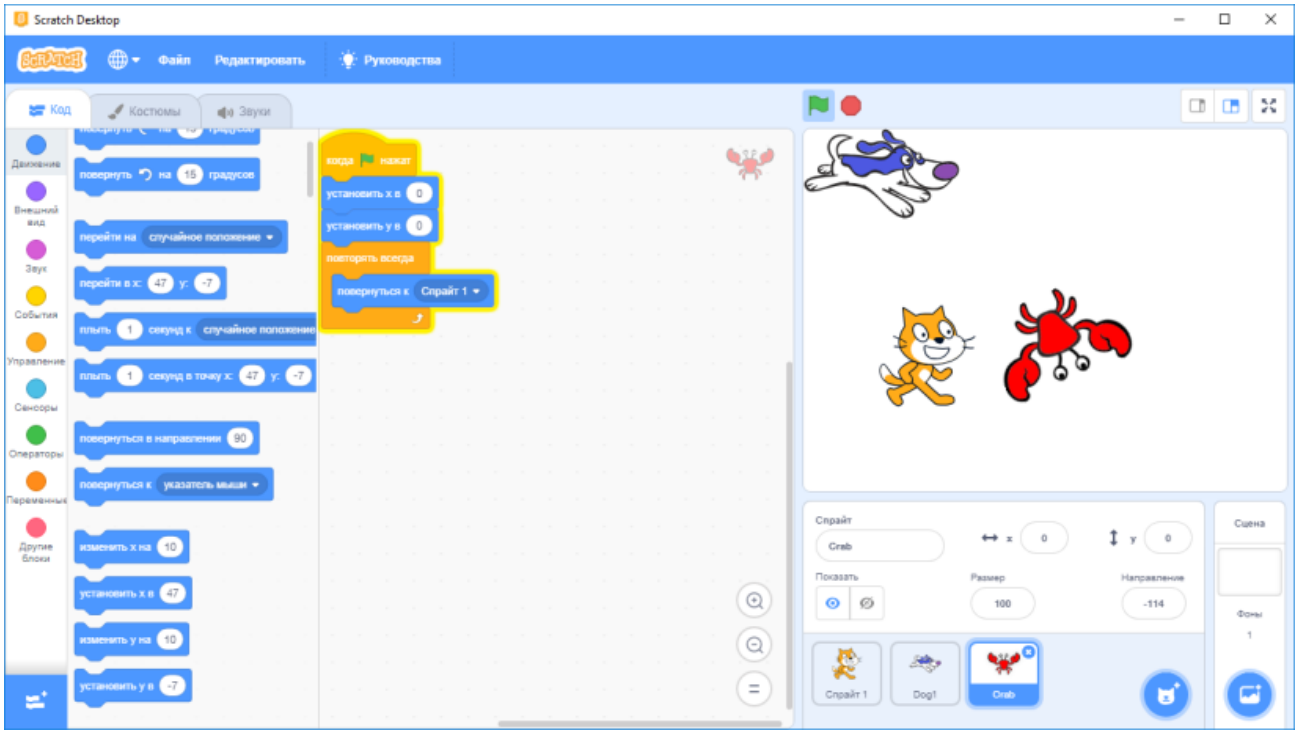
2.Какой цикл является вложенным, а какой внешним?

3.Какие команды содержит цикл *повтори ...*?

4.Какие команды содержит цикл *всегда*?

5.Сколько всего шагов делает кот, прежде чем останавливается на 3 секунды?

Добавим **третий** объект краб crub, но программу для него не будем копировать. Пусть третий объект стоит на месте в центре холста и всегда поворачивается ко второму объекту. Поскольку второй объект постоянно бегает по холсту, то и третий объект постоянно будет вертеться. Добавьте третий объект и соберите для него такой скрипт:



Команда *повернуться к ...* заставляет объект, к которому она применяется поворачиваться в направлении того объекта, который выбран в раскрывающемся списке этой команды. Цикл *всегда* здесь также необходим. Иначе объект повернется только один раз в самом начале. Поскольку второй объект у нас постоянно двигается, то и следить за ним надо всегда.

Нажмите кнопку запуска и посмотрите получившуюся анимацию. Все три объекта начинают свое движение одновременно, но двигаются по-разному, т.к. каждый из них управляется собственным скриптом.

Слои

Обратите внимание, какой из объектов при движении находится сверху, как бы перекрывает другие. Скорее всего, это будет третий объект. Второй объект при перемещении как бы проходит под ним и в тоже время, если вы заметили, находится выше первого. Получается, что на холсте есть как бы три слоя, и каждый объект двигается только по своему. Что же делать, если надо, чтобы третий объект находился ниже второго? Для этого достаточно взять второй объект (зажав левую кнопку мыши, когда курсор находится над ним) и положить его сверху третьего. Проделайте это и посмотрите анимацию. Отметьте, какой из объектов находится ниже. Остановите выполнение, теперь положите третий объект на второй. Снова запустите программу и отметьте разницу.

Следует знать, что перемещать объекты на другие слои на холсте можно не только вручную с помощью мыши, но и программно, когда в сценарий объекта встраиваются команды *перейти в верхний слой* и *перейти назад на … слоев*.

Разработки практических занятий

Одновременное выполнение скриптов (программ). Вспомните, как двигались три объекта на прошлом занятии. Они шли вместе, одновременно. Несмотря на то, что движение у каждого было свое (кот ходил слева направо, второй герой под углом в 45 градусов, а третий просто крутился на месте), они все равно начинали движение в одно и то же время — при нажатии на кнопку запуска. Это был пример, когда мы можем говорить об одновременном (параллельном) выполнении разных блоков команд (скриптов). В серьезном программировании используются другие, более умные слова (многопоточность и т.п.).

При этом в Scratch можно сделать так, что два (или больше) скрипта одновременно будет выполнять вообще один объект. Допустим, кот будет шагать и при этом менять свои размеры, форму и другие свойства. При этом ходьба и при щелчке на кнопке запуска начнут работать оба скрипта сразу. Первый будет заставлять кота ходить слева направо. Это нам уже знакомо. Второй скрипт будет менять внешний вид кота. Фиолетовые команды связаны с кнопкой **внешность.** Во втором скрипте выше используется команда *изменить ... эффект на ...*. После слова «изменить» в раскрывающемся списке можно выбрать понравившийся нам эффект, а в поле с числом прописать, на сколько единиц его изменять. Команда *убрать графические эффекты* возвращает объект к его исходному внешнему виду. Итак, второй скрипт в цикле *всегда* выполняет следующие действия: изменяет цвет объекта → оставляет объект в таком состоянии на 1 секунду → возвращает объект к прежнему цвету → искривляет объект с помощью завихрения → оставляет в таком состоянии на 1 секунду → возвращает к исходной форме. Составьте два подобных скрипта для одного объекта. Можете попробовать другие графические эффекты (например, рыбий глаз или мозаику).

Практическое занятие 1. Путешествие в космос

Тема: анимация объектов в среде программирования Scratch.

Цель: научиться анимировать объекты в среде программирования Scratch.

Оборудование: ПК с установленными ОС и Scratch, (данная) инструкция.

Структура урока:

1. Организационный момент.
2. Актуализация опорных знаний.
3. Изучение нового материала.
4. Инструктаж по ТБ.
5. Практическая работа.
6. Подведение итогов урока.
7. Домашнее задание.

Ход урока

1. Инструктаж по ТБ

2. Закрепление изученного материала

Примечание. После выполнения каждого из практических задач обучающийся должен сообщить учителю о своей готовности показать результаты своей работы.

3. Практическая работа проект «Путешествие в космос».

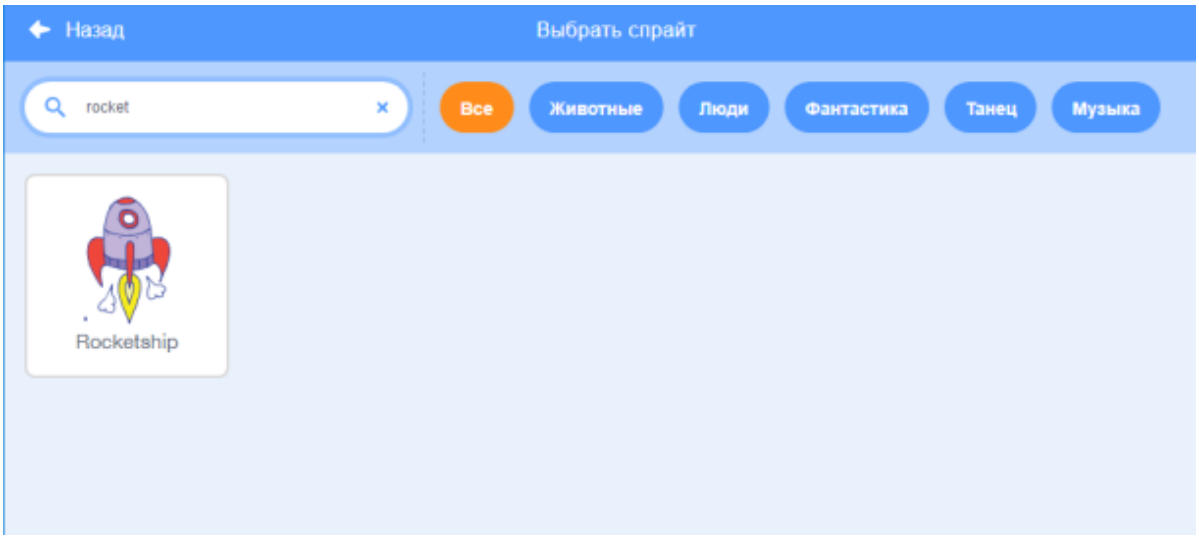
Шаг1. Подготовка

1. Создайте новый проект (меню Файл-Новый);
2. Удалите существующий спрайт (клик правой кнопкой мышки, удалить);
3. Добавьте новое тело (откройте библиотеку, изображение «Stars»);

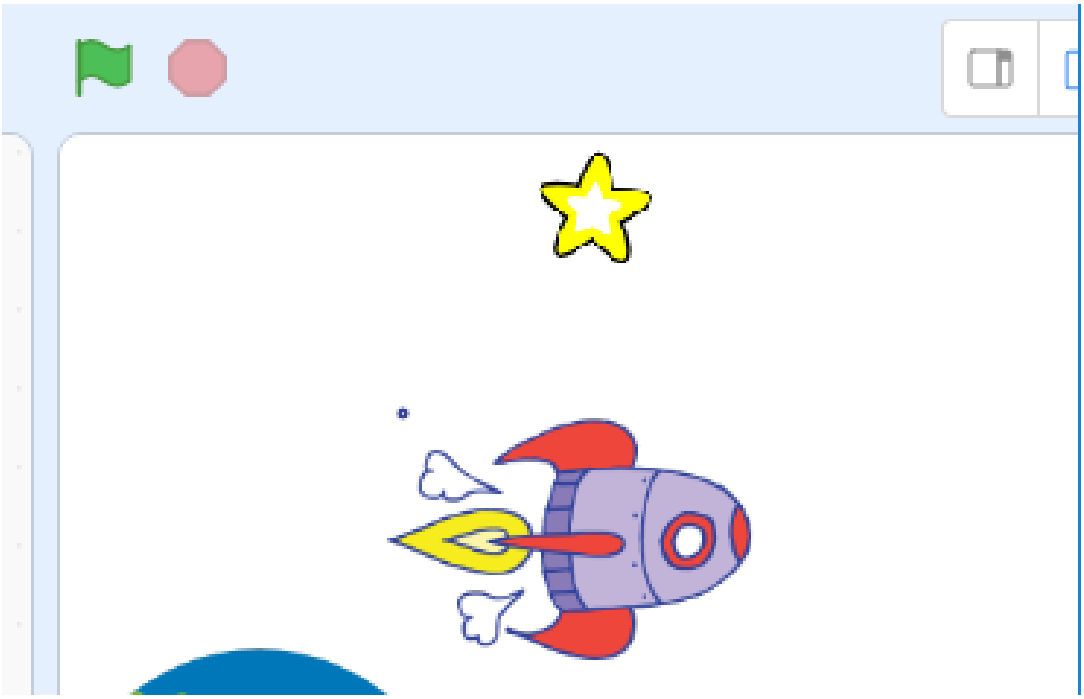
4. Удалите начальную сцену (клик правой кнопкой мыши, Удалить)

Шаг 2. Создание звездолета

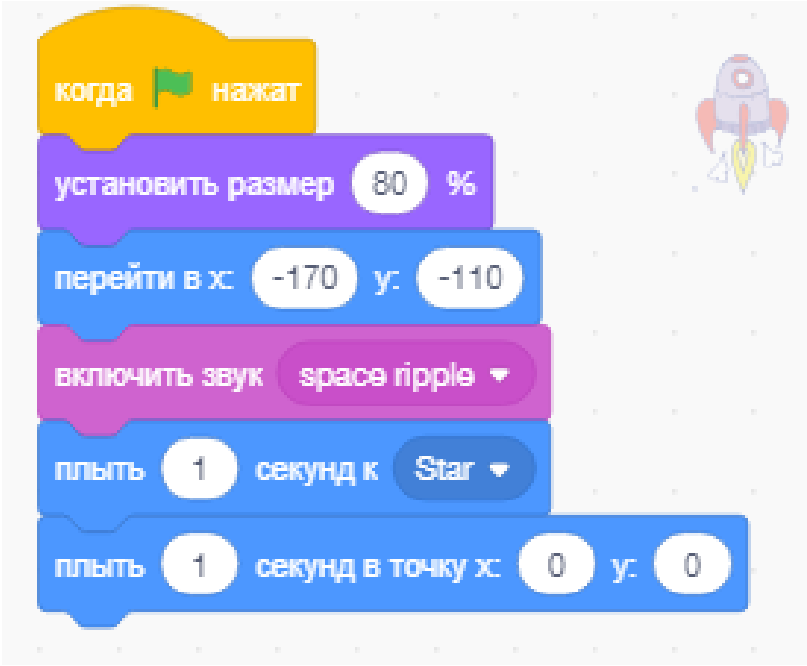
1. Добавьте новый спрайт звездолета (категория «Транспорт», спрайт «Rocketship»);



2. Перейдите на вкладку образы, выберите рисунок ракеты и поверните его так как показано на рисунке с помощью опции Направление.



3. Создайте скрипт, чтобы ракета находилась в левом нижнем углу и двигалась сначала к звезде, а потом в цент экрана со звуком полета.



Шаг 3. Создание Земли

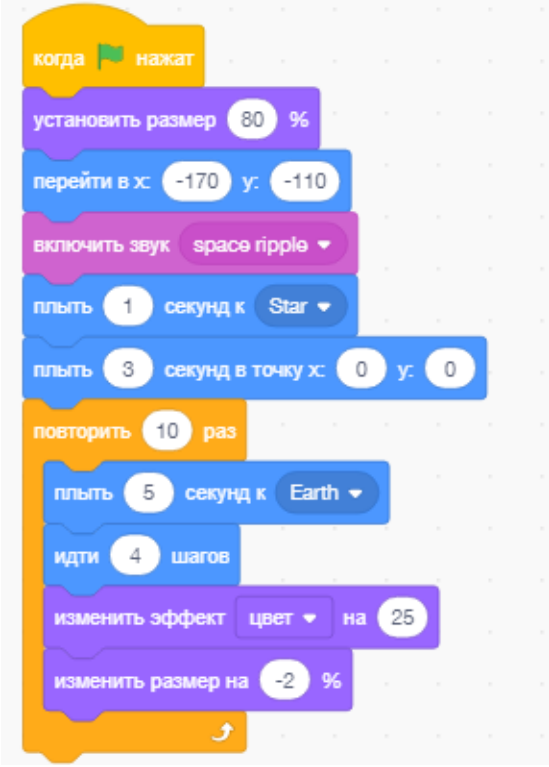
1. Добавьте новый спрайт (категория «Вещи» спрайт Земля «Earth»).

2. Создайте скрипт Земля, который перемещает спрайт в правый верхний угол сцены и устанавливает ее размер в 150%.

Шаг 4. Ракета летит на Землю

1. Добавьте команды, чтобы звездолет полетел на Землю.

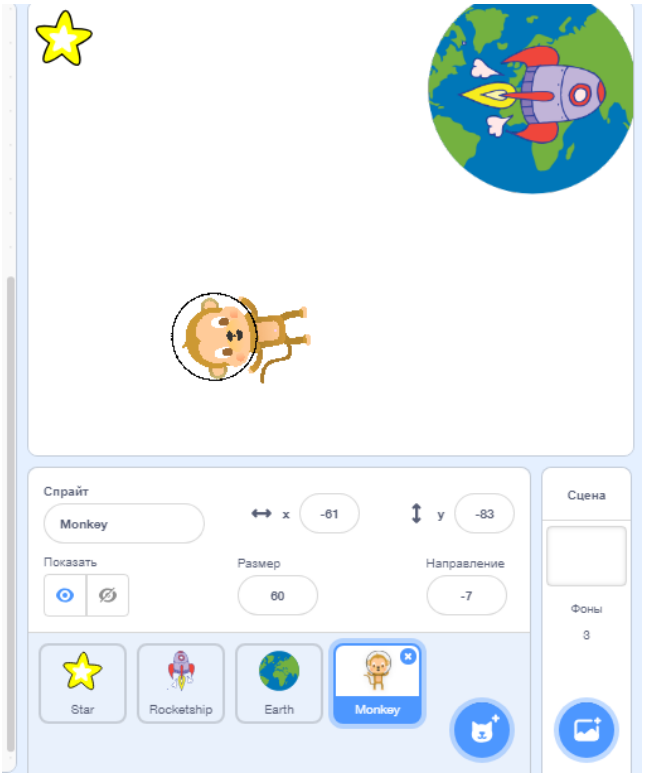
2. Добавьте анимацию уменьшения ракеты с расстоянием и изменением его цвета.



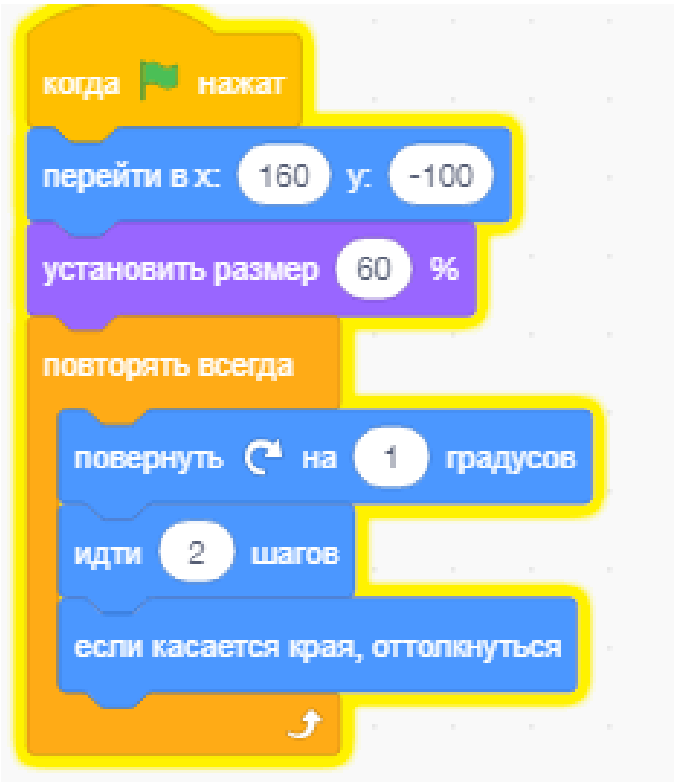
Шаг 5. Создайте спрайт Обезьянка-космонавт, которая потерялась в космосе

1. Добавьте новый спрайт обезьянка (категория «Животные», спрайт «Monkey1».

2. С помощью инструмента овал нарисуйте ему шлем.



3. Создайте скрипт, который будет показывать движение нашего космонавта в космосе.



Домашнее задание.

Создайте спрайт летучей мыши «Bat» и движение с уменьшением, как она улетает к звезде, можно добавить изменение цвета.

Практическое занятие 2. Рисование собственного спрайта

Тема: рисование собственного объекта в среде программирования Scratch.

Цель: научиться создавать и анимировать собственные объекты в среде программирования Scratch.

Оборудование: ПК с установленными ОС и Scratch, (данная) инструкция.

Структура урока:

1. Организационный момент.

2. Актуализация опорных знаний.

4. Инструктаж по ТБ.

5. Практическая работа

6. Подведение итогов урока.

7. Домашнее задание.

Ход урока

1. **Инструктаж по ТБ**

2. Закрепление изученного материала

Примечание. После выполнения каждого из практических задач обучающийся должен сообщить учителю о своей готовности показать результаты своей работы.

3. Практическая работа проект «Цветок».

1. Нарисовать объект: бутон - цветок, который еще не распустилась.

2.Добавить этому объекту образы, изображающие стадии раскрытия цветка от бутона до полностью раскрывшегося цветка. Названия образов - слово (предпочтениям учащихся) + № стадии распускания цветка.

3. Создать анимацию - раскрытие цветка.

4. Нарисовать объект-солнце.

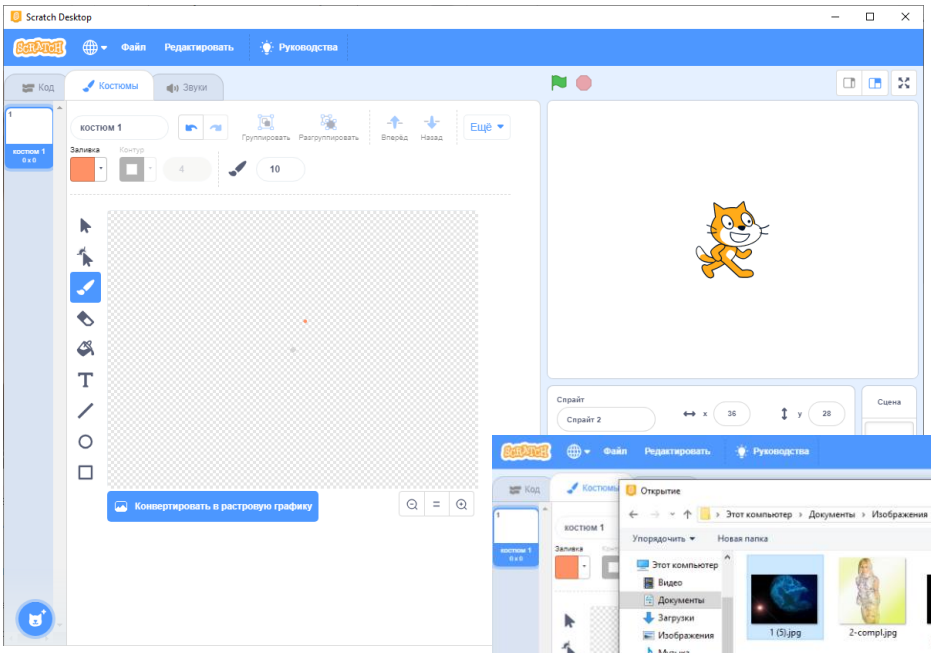
5.Создать анимацию - восход солнца (распускания цветка происходит утром).

Примечание.

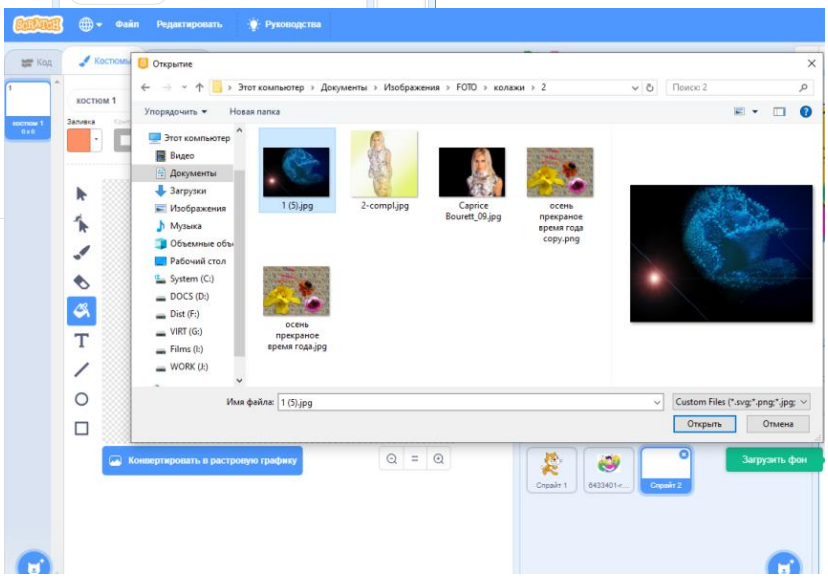
Создание собственных анимированных объектов - это сильный мотив для детей, ориентированных на художественное творчество. Для них сторона программирования может оказаться сложной. Нужно форсировать их развитие в области собственно программирования. Инструменты редактора можно рассматривать безотносительно к проектированию.

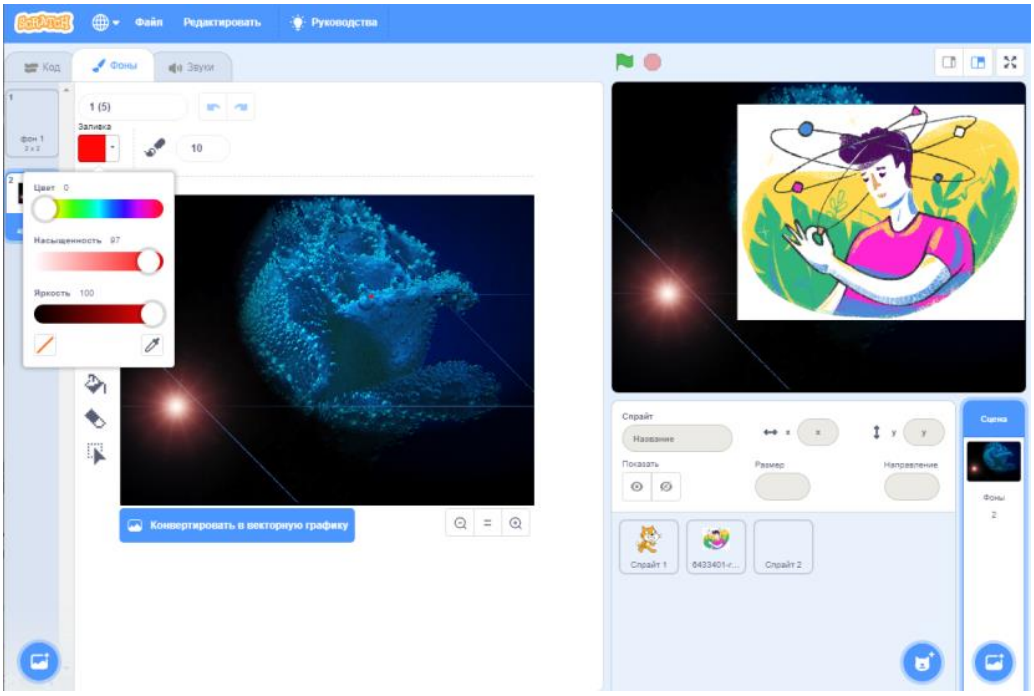
Выполнение задания 1.

1. Выбрать сцену в правой нижней части программного окна, перейти на закладку Фоны и нажать кнопку Импортировать новый фон.



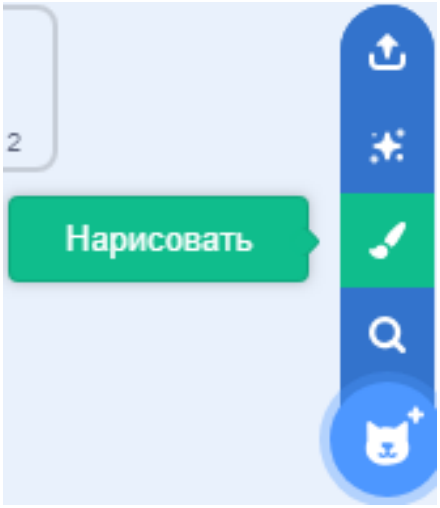
2. Появится диалоговое окно Импортировать фон. Выбрать и вставить фон.



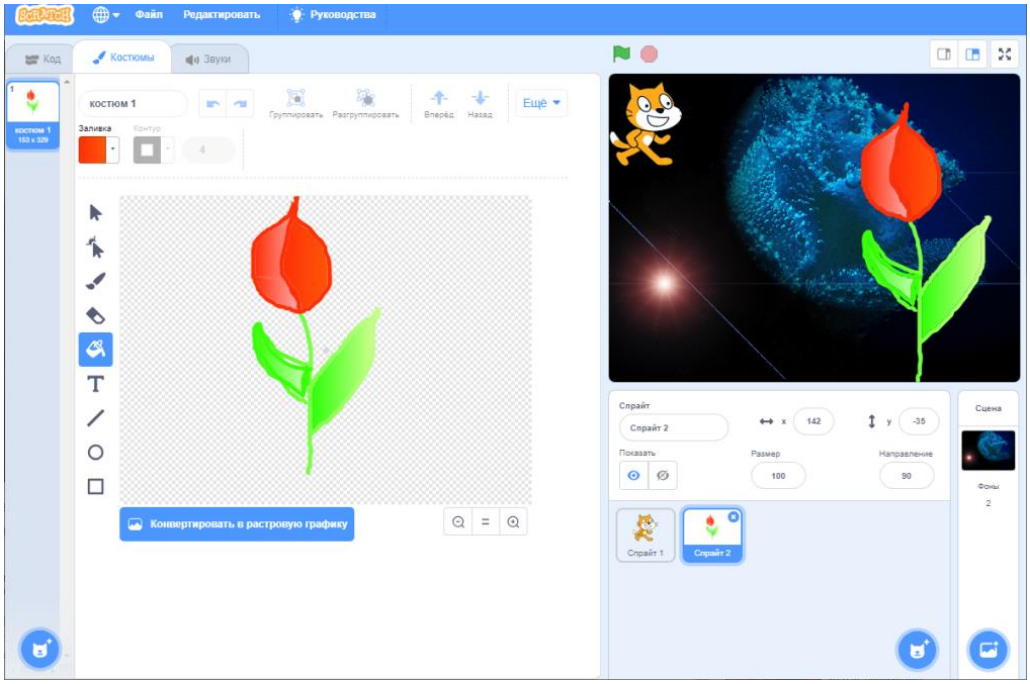


4. В этом окне выбрать категорию изображений, а среди них - изображение и нажать кнопку ОК.

5. Создать новый объект Цветок. Для этого нажать кнопку

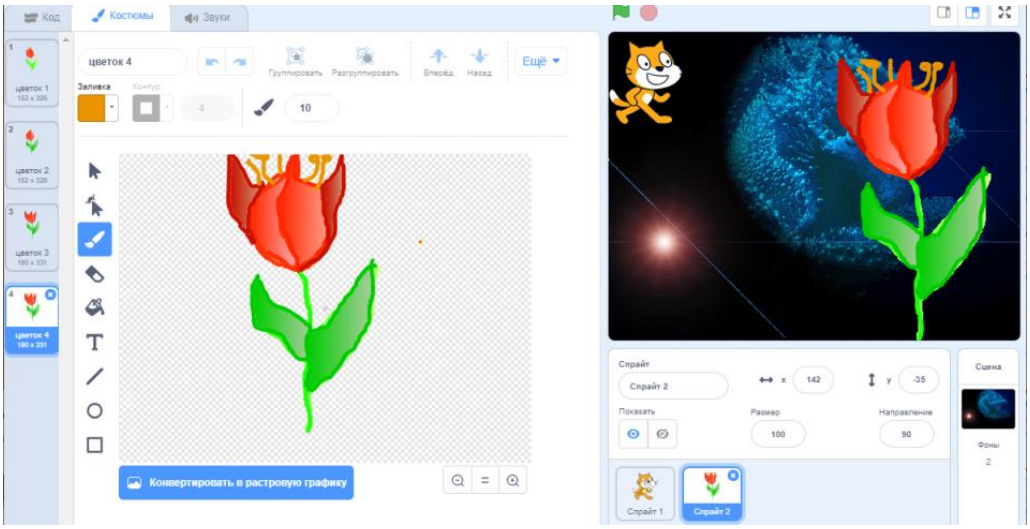


6. Нарисовать новый объект. В вызванном окне Графический редактор нарисовать первый образ цветка - бутон. Затем нажать кнопку ОК.

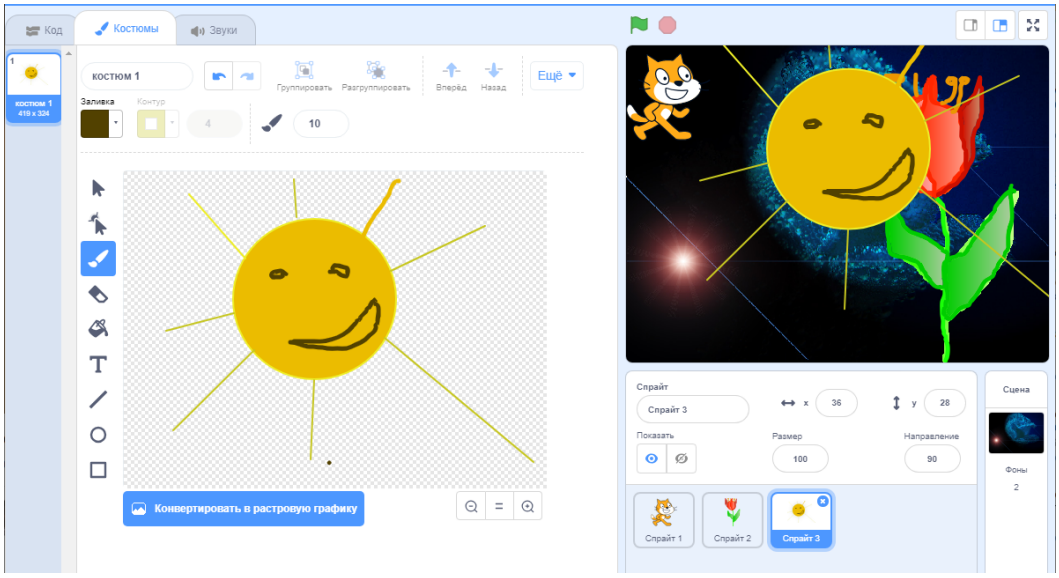


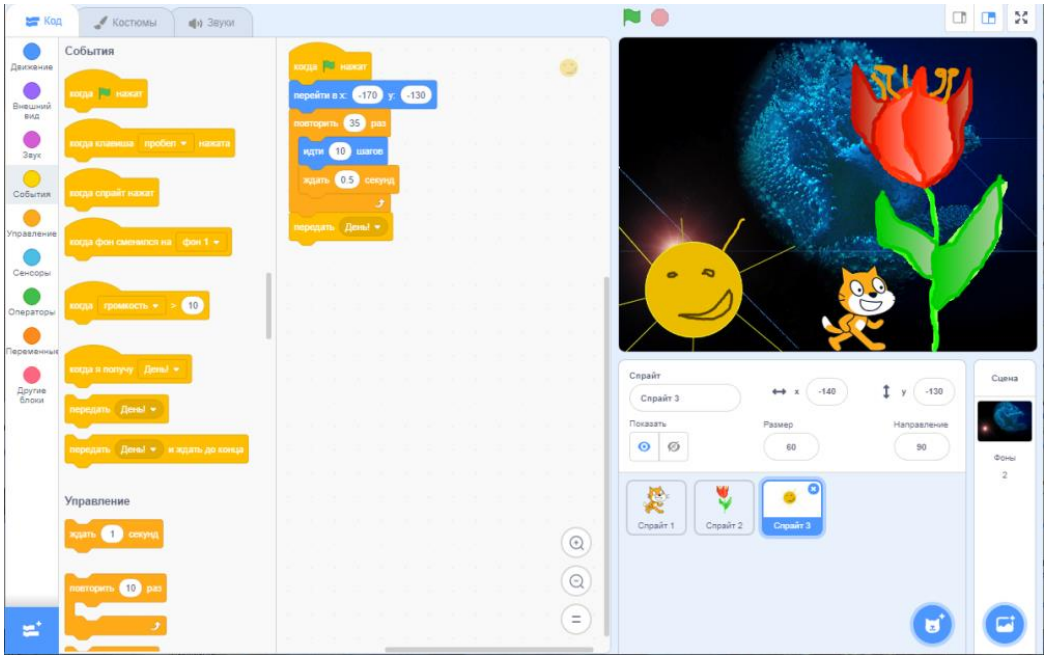
Созданный образ будет называться Костюм 1. Его нужно переименовать в Цветок 1. А также необходимо переименовать новый объект на Цветок. Название объекта записано в верхней части окна.

7. Дублировать образ Цветок 1, нажав кнопку Копировать. На новом образе Цветок 2 нажать кнопку Редактировать и в окне графического редактора нарисовать следующую стадию раскрытия цветка. Повторить действия несколько раз и нарисовать 4 образазараза.

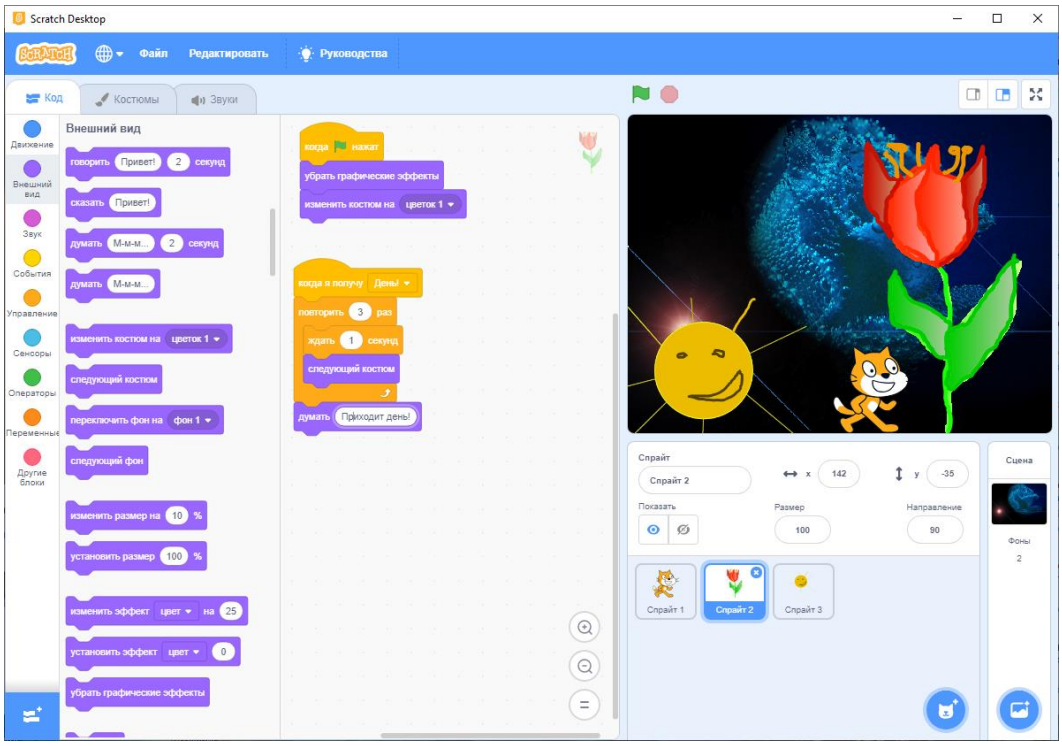


8. Создать новый объект Солнце, но нарисовать только один образ Солнце. Для реализации восхода этого объекта строго по вертикали, а по диагонали, ему нужно соответствующим образом задать (изменить) направление движения - схватить мышью синий вектор направления на информационной панели и повернуть его против часовой стрелки. Как устойчиво направление движения объекта - справа.





9. Аналогично создать скрипты для объекта Цветок, как показано на рисунке ниже.



***6.* Подведение итогов урока.**

***7.* Домашнее задание.**

Придумайте движение для кота

Практическое занятие 3.Ппроект «Часы»

Тема: Рисование и программирование в среде Scratch.

Цель: научиться создавать и анимировать собственные объекты в среде программирования Scratch.

Оборудование: ПК с установленными ОС и Scratch, (данная) инструкция.

Структура урока:

1. Организационный момент.
2. Актуализация опорных знаний.
3. Инструктаж по ТБ.
4. Практическая работа.
5. Подведение итогов урока.
6. Домашнее задание.

Ход урока

Работаем вместе.

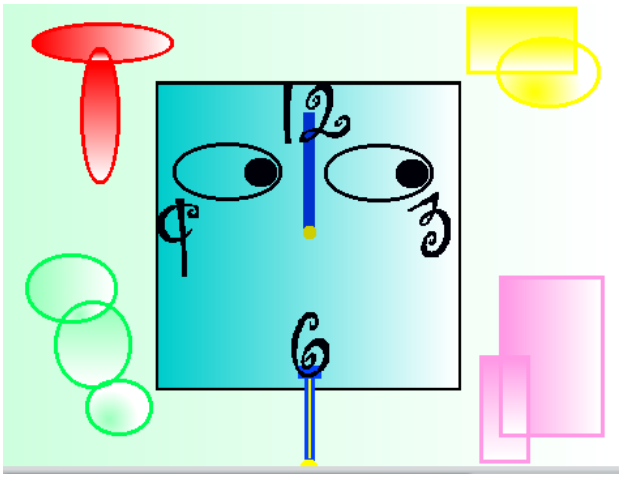
1. Нарисовать объект Часовая стрелка. Предоставить этому объекту еще три образа.

2. Нарисовать объект Минутная стрелка. Предоставить этому объекту еще три образа.

3. Нарисовать объект Глаза. Предоставить этому объекту еще один образ.

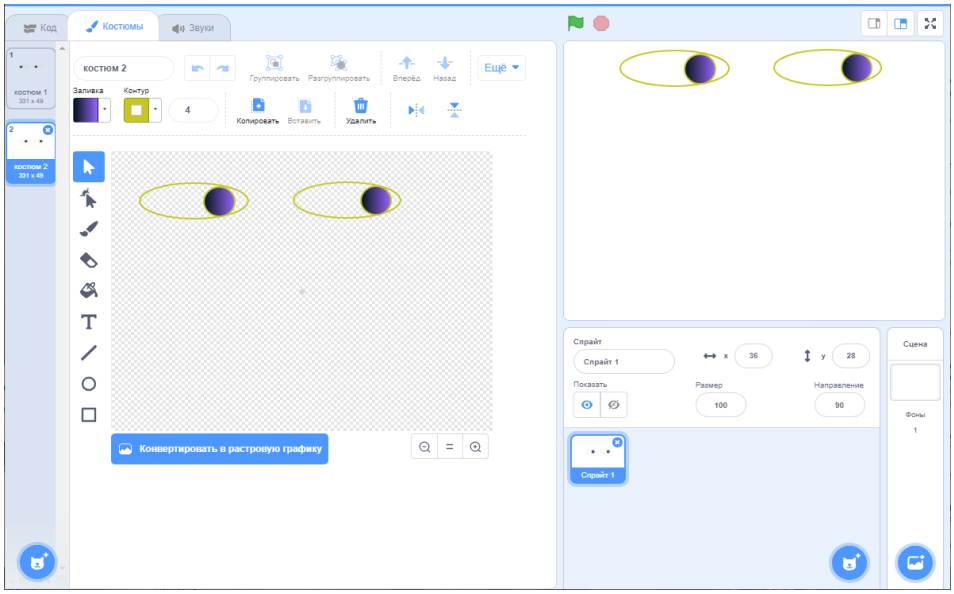
4. Провести анимацию последовательно меняя образы объектов с небольшой задержкой.

Общий вид проекта должен быть таким.

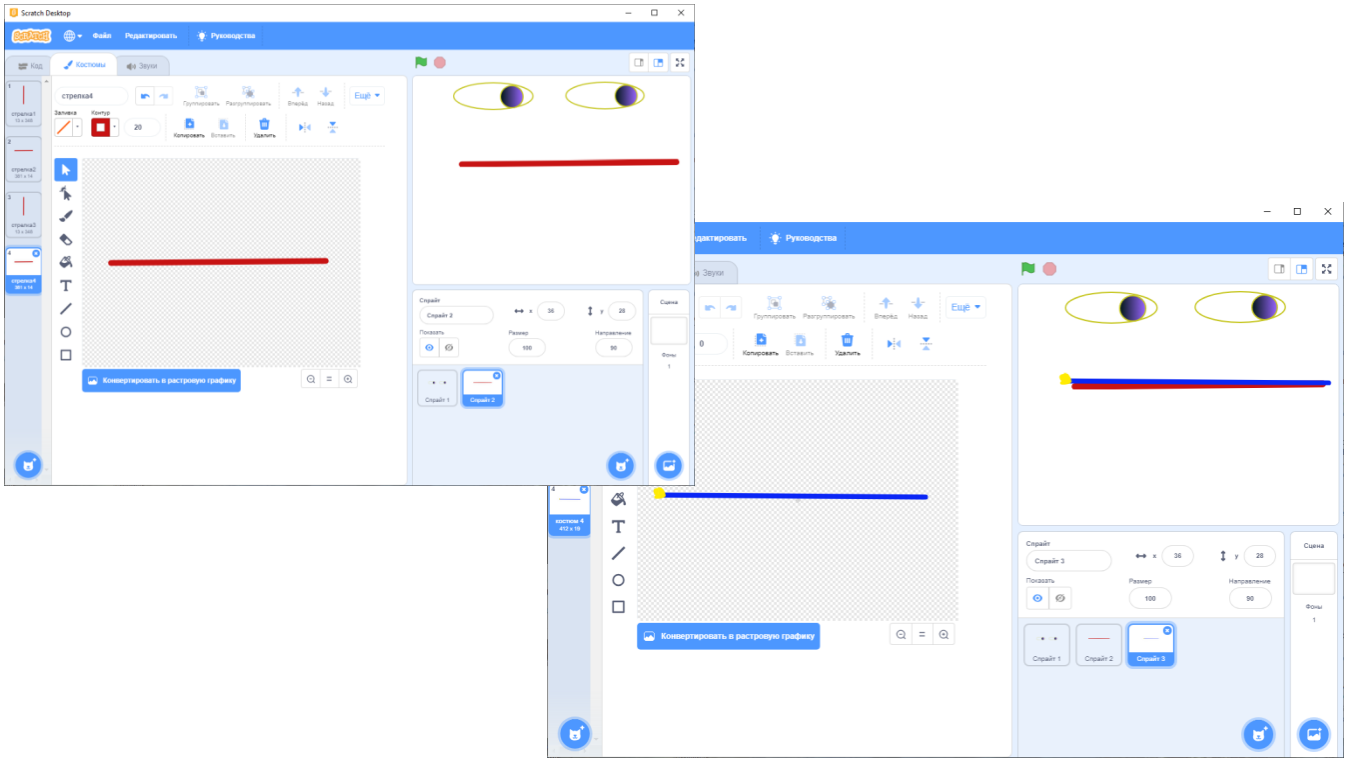


Выполнение задания.

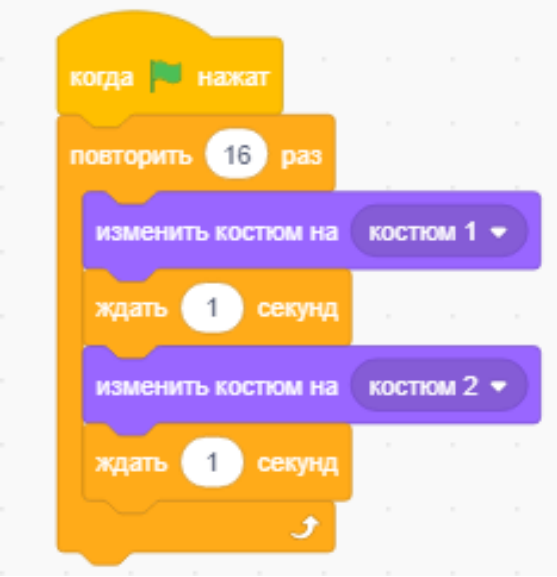
1. Сначала изменить фон Сцены. Затем нарисовать объект Глаза, которые будут двигаться в такт со стрелками.



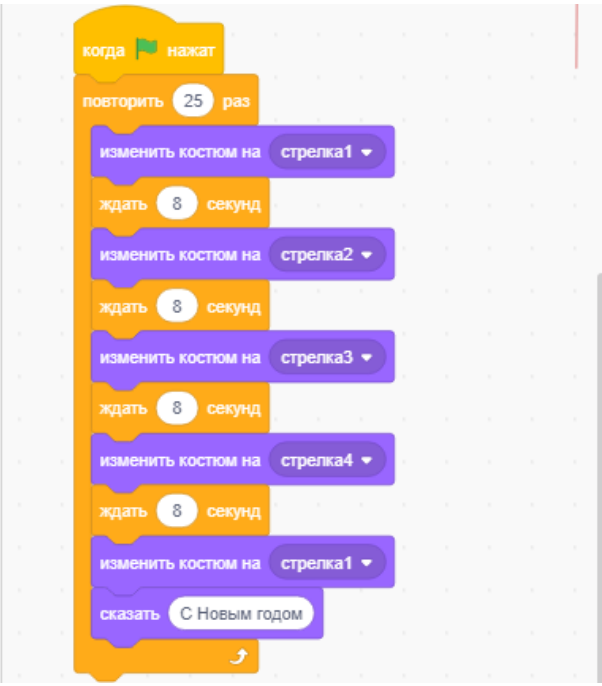
2. Часовую стрелку с образам.



3. Создать скрипты для глаз.



4. Создать скрипты для часовой стрелки и минутной стрелки.



5. В указаниях для стрелок нужно задать разную задержку движения стрелок.

Подведение итогов урока. Выставления оценок.

Домашнее задание: В проекте «Часы» дорисовать интерьер квартиры и маятник, который будет двигаться в такт со стрелками.

2.3 Апробация методики обучения школьников программированию в среде разработки Scratch

Мы проведем апробацию с целью определения того, что программа курса в Scratch, будет являться хорошим введением в тему «Начало программирования».

Идея апробации заключается в том, чтобы показать позитивное влияние практических занятий в Scratch на развитие алгоритмических умений у обучающихся. Апробация направлена и на решение педагогических задач, в частности, на развитие алгоритмического мышления у обучающихся посредством практических работ. В связи с этим *цель планируемой пробации:* доказать, что программа курса является хорошим введением в тему «Начало программирования» и способствует развитию алгоритмических умений у обучающихся.

Для реализации цели апробации выдвинуты следующие задачи:

1. Продиагностировать у обучающихся уровень развития алгоритмических умений.

2. Организовать занятия с использованием программы курса по изучению Scratch.

3. Проанализировать результаты апробации.

Апробация планируется в три этапа:

I. Подготовительный

II. Рабочий

III. Аналитический

В процессе изучения главы «Основы алгоритмизации» обучающиеся должны освоить следующие умения: разрабатывать план решения задачи, выдвигать и доказывать гипотезы, прогнозировать результаты решения, анализировать и находить рациональные способы. Именно эти мыслительные умения характеризуют уровень развития алгоритмического мышления.

Алгоритмическое мышление – совокупность мыслительных действий и приемов, нацеленных на решение задач, в результате которых создается алгоритм, являющийся специфическим продуктом человеческой деятельности.

Под способностью алгоритмически мыслить понимается умение решать задачи различного происхождения, требующие составления плана действий для достижения желаемого результата.

Поэтому на подготовительном этапе деятельность была направлена не только на разработку программы курса, но и подборку диагностического инструментария для осуществления оценки сформированности алгоритмических умений.

Была разработана диагностическая работа, целью которой было определение первоначального уровня развития алгоритмических умений у обучающихся. Работа состояла из 5 заданий. (Приложение 1).

В качестве критериев для оценки выполнения заданий были приняты следующие умения, характеризующие уровень сформированности алгоритмического мышления:

- умение упорядочивать действия в алгоритме;
-умение выполнять алгоритм: линейный, разветвленный;
- умение составлять план действий;
- умение сопоставить задачу и готовый алгоритм – модель ее решения.

Правильно выполненное задание оценивалось 1 баллом. Максимальное количество баллов – 5.

Рабочий этап был осуществлен на базе МАОУ СОШ №7 с. Патруши. Обязательным требованием к выбору базовой площадки для апробации выступало наличие компьютерного класса в ОУ и реализация курса информатики и ИКТ в соответствии с ФГОС ООО.

Работа была организована в 8б классе. Группа респондентов составила 21 человек.

Классный коллектив стабильный, самостоятельный, в процессе наблюдения сплоченность коллектива проявлялась на низком уровне. Общий уровень дисциплины в классе - удовлетворительный.

Для успешной организации апробации необходимо соблюдение социально – педагогических и психолого-дидактических условий.

Социально – педагогические условия:

1) необходимые средства обучения – компьютер, проектор;

2) доверительность отношений между учащимися и учителем, основанных на взаимном уважении и интерактивности, что обеспечит благоприятную обстановку в «учебной» аудитории, способствующую успешному овладению учебным содержанием;

3) использование разнообразных методов обучения:

- словесные методы обучения (рассказ, объяснение, лекция, беседа, работа с учебником и книгой);

- наглядные методы (наблюдение, иллюстрация, демонстрация презентаций);

- практические методы (устные и письменные упражнения, практические компьютерные работы).

Социально – дидактические:

1) статус школы (зависит от того, какое оборудование есть в информационных классах, по какой программе работают учащиеся, компетенции учителя);

2) уровень обученности обучающихся, соответствующий программным требованиям, основанный на логичности, преемственности и последовательности содержания;

3) соблюдение дидактических принципов и правил организации учебного процесса, так как в условиях гуманизации образования, развития новых экономических механизмов вся совокупность требований к учебному процессу сводится, прежде всего, к соблюдению дидактических принципов обучения;

4) наличие у обучающихся возможности использовать компьютер в домашних условиях.

На первом уроке была проведена диагностическая работа.

Результаты диагностической работы представлены в таблице (Приложение 2).

На основании этих результатов был проведен анализ выполнения заданий и получены следующие выводы.

С заданием № 1, в котором требовалось упорядочить действия в алгоритме, успешно справились 86% обучающихся – все действия поставлены в правильном порядке.

С заданием № 2, на выполнение линейного алгоритма, справились 90% обучающихся – они точно выполнили все предписания алгоритма, 3 обучающихся не справились, показав непонимание требуемых предписаний алгоритма.

С заданием № 3, на выполнение разветвленного алгоритма, справились 57 % обучающихся, у остальных возникли трудности при вычисление значения переменной x.

С заданием № 4, в котором проверялось умение составлять план действий, успешно справились 67 % обучающихся – все действия поставлены в правильном порядке.

Задание № 5, на сопоставление задачи и готового алгоритма, успешно справились 95% обучающихся. Не смог выполнить задание всего 1 обучающийся.

Результаты выполнения отдельных заданий представлены на рисунке 4.

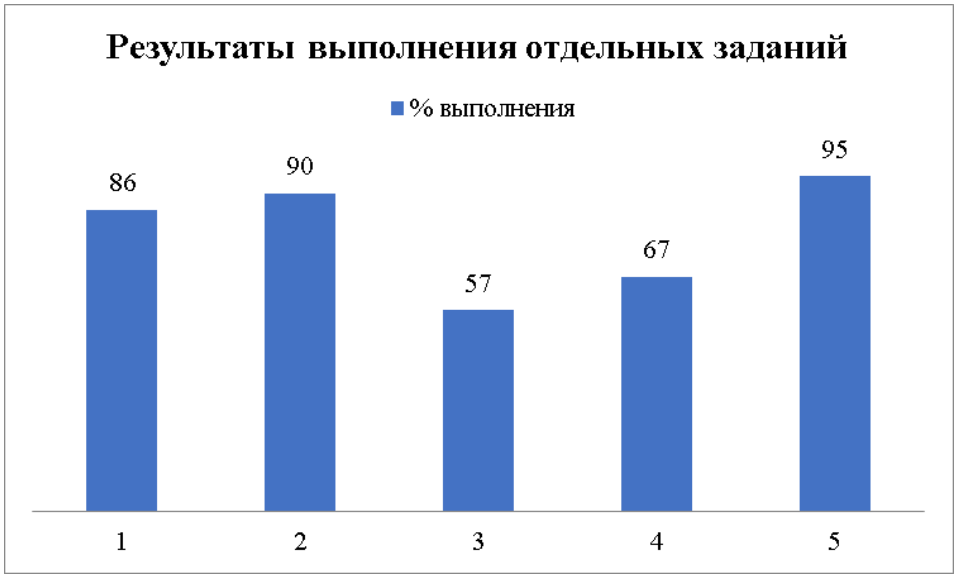


Рисунок 4. Результаты выполнения отдельных заданий

По результатам выполнения заданий были выделены следующие уровни:

- 90-100% – высокий уровень;
- 65-90 – средний уровень;
- менее 65 – низкий уровень.

Распределение обучающихся по уровням сформированности алгоритмических умений (рисунок 5).

Рисунок 5. Распределение обучающихся по уровням сформированности алгоритмических умений

В ходе первичного замера было установлено, что в классе, в целом, наиболее сформированы такие показатели (Приложение 2), как: упорядочивать действия в алгоритме; выполнять линейный алгоритм; сопоставить задачу и готовый алгоритм – модель ее решения.

Это обусловлено тем, что обучающиеся уже имели опыт выполнения подобных заданий.

Таким образом, на рабочем этапе был выявлен уровень сформированности алгоритмических умений у обучающихся 8б класса Патрушевской школы №7.

После оценки сформированности алгоритмических умений началась апробация курса. Было отведено 4 урока в рамках учебного процесса.

Были проведены следующие уроки:

3. *Урок лекция*. Интерфейс Scratch.
4. *Практическое занятие:* Путешествие в космосе.
5. *Практическое занятие:* Рисование собственного спрайта.
6. *Практическое занятие:* Проект «Часы».

На этапе знакомства с программой Scrath, работа была организована во фронтальном режиме (учитель показывал, знакомил с интерфейсом и основными инструментами).

Уроки прошли успешно, обучающимся нравилось заниматься в среде программирования Scratch, материал был освоен хорошо. Дети с интересом выполняли практические работы. На первых уроках ученики не могли запомнить значения определенных блоков, терялись в интерфейсе. Но спустя 1 занятие, школьники освоили интерфейс программы. Так же, ученики начали придумывать свои собственные истории и воплощать их на экранах компьютера с помощью Scratch дома.

Целью *аналитического этапа* выступает анализ проделанной работы на рабочем этапе.

В ходе первичного замера было установлено, что умения 3 и 4 находятся на низком уровне. Поэтому вторичный замер (Приложение 3) был ориентирован на данные умения.

В ходе вторичного замера мы получили следующие результаты. (Приложение 4), (Рисунок 6).

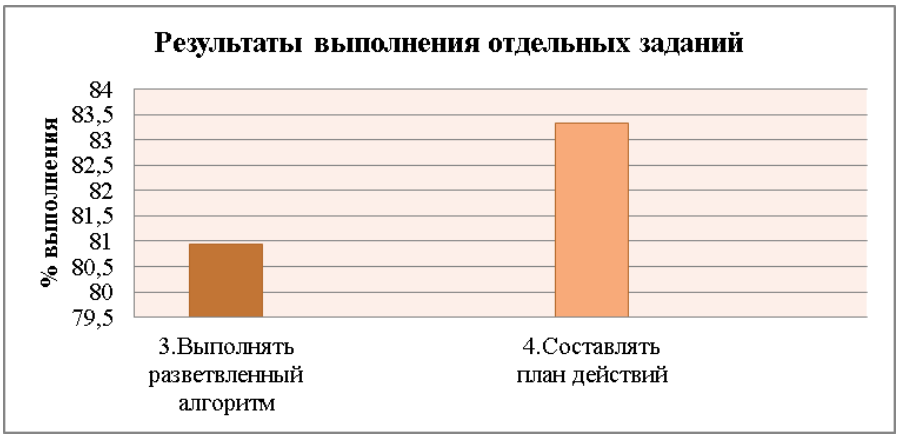


Рисунок 6. Результаты выполнения отдельных заданий

С умением № 3, на выполнение разветвленного алгоритма, справились 81 % обучающихся. С умением № 4, в котором проверялось умение составлять план действий, успешно справились 83 % обучающихся.

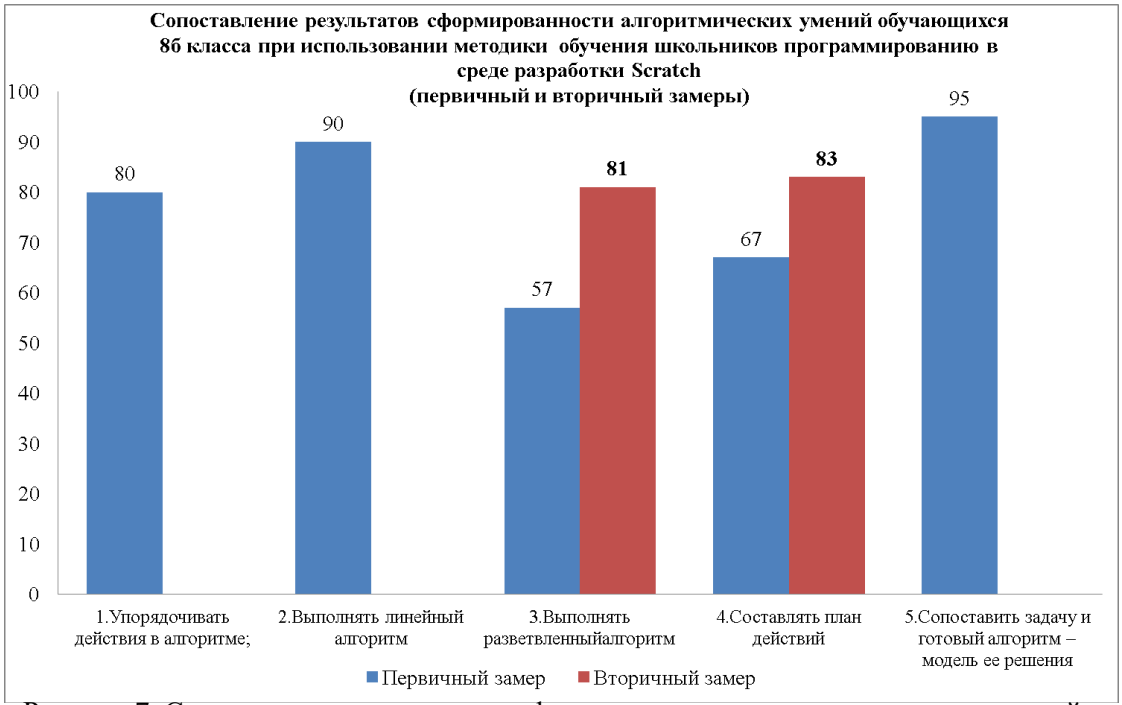


Рисунок 7. Сопоставление результатов сформированности алгоритмических умений

Анализируя результаты первичного и вторичного замеров, мы делаем вывод, что по данным показателям произошли существенные изменения (Рисунок 7).

Распределение обучающихся по уровням сформированности алгоритмических умений при использовании методики обучения школьников программированию в среде Scratch (Рисунок 8).

Рисунок 8. Распределение обучающихся по уровням сформированности алгоритмических умений. (вторичный замер)

Положительный результат апробации обусловлен четко составленными практическими занятиями, понятными для данной возрастной группы, а также красочным и понятным интерфейсом программы Scratch, вызывающим интерес у школьников. Что указывает на достаточный уровень разработанности программы курса по изучению Scratch в 8 классе.

По результатам пробации можно определить перспективные направления деятельности в части разрабатываемой проблемы:

1. Совершенствование методических рекомендаций по организации курса, дополнения примерами работ обучающихся и алгоритмами выполнения некоторых типовых заданий.

2. Совершенствование содержания конспектов занятий за счет расширения спектра сюжетов занятий.

Таким образом, проведенная апробация позволила организовать занятия в 8б классе на основе разработанной методики и установить ее эффективность в части развития у обучающихся алгоритмических умений. Все это позволяет сделать вывод о том, что поставленные задачи и цель исследования достигнуты в полном объеме.

ЗАКЛЮЧЕНИЕ

Подводя итоги проделанной работы, можно сделать вывод о том, что цель исследования была достигнута, нами был разработан курс обучения программированию в Scratch.

Курс разработан на основе проанализированных программ, моделирующих процесс программирования в наглядно-графической форме, рассчитанный на обучающихся 8 класса.

Мы поставили и решили следующие задачи:

1. Определили основные достоинства и преимущества использования Scratch в процессе изучения основ и принципов программирования обучающимися общеобразовательной школы на уроках информатики.
2. Проанализировали существующие программы, моделирующие процесс программирования в наглядно-графической форме.
3. Разработали и обосновали систему занятий для обучения программированию школьников с использованием визуальной среды разработки Scratch;
4. Провели апробацию разработанных материалов, которая показала заинтересованность в изучении программирования в среде Scratch.

В ходе нашей работы мы определили, что Scratch, является хорошим введением в тему «Начало программирования», а так же оказывает позитивное влияние на развитие алгоритмических умений у обучающихся.

В результате проведенной работы школьники, завершившие изучение курса, продолжают использовать Scratch, в частности, в процессе организации проектной деятельности. При этом создают проекты по собственному желанию, что указывает на целесообразность методических разработок по работе в Scratch-среде.

Приобретенные школьниками приемы работы и умение использовать алгоритмические конструкции дают возможность более основательно и самостоятельно работать с программой. Использование среды Scratch в учебной деятельности позволяет: активизировать процесс их подготовки к учебным занятиям, созданию интерактивных анимированных материалов, коллективно работать над проектами и обмениваться результатами через Scratch сообщество, побуждать к самостоятельной деятельности..

СПИСОК ЛИТЕРАТУРЫ

1. Денисова Л.В. Проектная деятельность школьника в среде программирования Scratch / Л.В. Денисова, В.А. Дженжер, В. Рындак // Оренбург. 2009. Режим доступа: https://sites.google.com/site/ orenscratch / nasirazrabotki. - Название с экрана.
2. Денисова Л.В. Среда Scratch в практике учителя начальной школы / Л.В. Денисова, В.О.Дженжер // Начальная школа. 2012. - № 5. С. 31-35.
3. Кончакова Р.Б. Анимационные программы в курсе информатики для студентов гуманитарных специальностей / Р.Б. Кончакова, М.Ю. Сидляр // Психологопедагогический журнал Гаудеамус. 2014 - № 2 (24). - С. 137-143.
4. Лесневский А. С. Объектно-ориентированное программирование для начинающих / А. С. Лесневский. – М. : БИНОМ. Лаборатория базовых знаний, 2005. – 232 с.
5. Пейперт С. Переворот в сознании: дети, компьютеры и плодотворные идеи / Сеймур Пейперт. – М. : Педагогика, 1989. – 224 с. 10
6. Ломаковская Г. В. Ступеньки к информатике: учеб. для 2 кл. общеобразоват. учеб. заведений / Г. В. Ломаковская, Г. А. Проценко, И. Я. Ривкинд, Ф. М. Ривкинд. - М.: Издательский дом «Образование», 2012. - 160 с.
7. Ломаковская Г. В. Ступеньки к информатике: учеб. для 3 кл. общеобразоват. учеб. заведений / Г. В. Ломаковская, Г. А. Проценко, И. Я. Ривкинд, Ф. М. Ривкинд. - М.: Издательский дом «Образование», 2013. - 160 с.
8. Патаракин Е.Д. Педагогический дизайн социальной сети Scratch / Е.Д. Патаракин // Образовательные технологии и общество (Educational Technology & Society). 2013. - № 2. - С.505-528.
9. Зайцев В.В. Современные педагогические технологии: учебное пособие/ В.В. Зайцев//Челябинск, ЧГПУ, 2012-411с.

10. Патаракин Е.Д. Учимся готовить в среде Скретч / Е.Д. Патарикин. - М.: 2007. - Режим доступа: http://umr.rcokoit.ru/dld/metodsupport/scratch1.pdf. - Название с экрана.
11. Патаракин Е.Д. Школа Scratch / Е.Д. Патарикин // Школьные технологии. 2010. - № 4. - С. 132 - 135.
12. Сорокина Т.Е. Визуальная среда Scratch как средство мотивации учащихся основной школы к изучению программирования / Т.Е. Сорокина // Информатика и образование. 2015 - №5 (264). - С. 30 - 34.
13. Теплицкий А. И. Средства обучения объектно-ориентированного моделирования студентов естественных специальностей педагогических университетов / А. И. Теплицкий // Сб. наук. пр. Кам.-Подол. нац. ун-та. Серия педагогическая. - Каменец-Подольский: Кам.-Подол. нац. ун-т им.И.ОГИЕНКА, 2011. - Вып. 17. - С. 246-248.
14. Шлянчак С.А. Использование социальных сервисов интернет в учебной деятельности студентов / С. А. Шлянчак // Информационные технологии в образовании. 2016. - № 3 (28). С. 84-93. - Режим доступа: http://ite.kspu.edu/webfm_send/901. - Название с экрана.
15. Босова Л. Л. Информатика : учебник для 6 класса / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ. Лаборатория знаний, 2013.– 213 с. : ил.
16. Босова Л. Л. Информатика : учебник для 8 класса / Л. Л. Босова, А. Ю. Босова. – 3-е изд. – М. : БИНОМ. Лаборатория знаний, 2015.– 160 с. : ил.
17. Босова Л. Л., Методика применения интерактивных сред для обучения младших школьников программированию / Л. Л. Босова, Т. Е. Сорокина // Информатика и образование. – № 7 (256). – 2014.
18. Сорокина Т. Е. Визуальная среда Scratch как средство мотивации учащихся основной школы к изучению программирования // Информатика и образование. – № 5 (264). – 2015.

19. Сорокина Т. Е. Методика раннего общедоступного программирования в основной образовательной программе. Сборник научных трудов XI Международной научно-практической конференции «Современные информационные технологии и ИТ-образование». – 2016. Т. 12. № 3-1. – С. 228–232.
20. Тихомирова Л.Ф. Развитие логического мышления детей. / Л.Ф. Тихомирова, А.В. Басов. – Ярославль: ТОО “Тринго”, 1995. – 235 с.
21. Овчинникова Т. Н. Личность и мышление ребенка: диагностика и коррекция / Т. Н.Т Овчинникова. – М.: Академический Проект, 2001. – 192 с.
22. Касьянова Т.В. Развитие логического, алгоритмического мышления и творческих способностей. [Электронный ресурс] / режим доступа: http://nsportal.ru/shkola/obshchepedagogicheskie tekhnologii/library/2013/10/07/uchitelskiy-proekt-razvitie
23. Еремеева Н.Н. Формирование алгоритмического мышления у школьников в ходе групповой работы. / Еремеева Н.Н. // Пермский педагогический журнал – 2013. – № 4.– С. 25-29.
24. Ершов А. П. Школьная информатика: (концепции, состояние, перспективы) / А. П.Ершов, Г. А. Звенигородский, Ю. А. Первин. – Новосибирск: Препринт ВЦ СО АН СССР, 1979. – 51 с.
25. Вершинин О.Е. За страницами учебника информатики. / О.Е. Вершинин. – М., 1992. – 352 с.
26. Газейкипа А. И. Стили мышления и обучение // Информационные технологии в общеобразовательной школе. – 2003. – С. 12-19.
27. Нежинская О.И. Логика. 1 класс. Занимательные упражнения для развития логического мышления. / О.И. Нежинская. – Волгоград: Учитель– АСТ, 2005. – 96 с.

28. Козлова Е.Г. О возможностях формирования у младших школьников способности к работе с алгоритмизированными обучающими средствами / Е.Г. Козлова. // Начальная школа. – 2004. – № 2. – С. 99-112.
29. Лапчик М.П. Вычисления. Алгоритмизация. Программирование. / М.П. Лапчик. – М., 1988. – 207 с.
30. Лебедева Т. Н. Формирование алгоритмического мышления школьников в процессе обучения рекурсивным алгоритмам в профильных классах средней общеобразовательной школы: дис. ... канд. пед. наук: 13.00.02. – Екатеринбург, 2005. – 219 с.
31. Леонтьев А.Н. Проблемы развития психики / А.Н. Леонтьев. – М.: Изд-во МГУ, 1981. – 584 с.
32. Издательство «БИНОМ. Лаборатория знаний» [Электронный ресурс]/ режим доступа: http://www.lbz.ru/metodist/iumk/informatics/files/bosova-7-9-prog.pdf
33. Патаракин Е.Д. Школа Scratch / Е.Д. Патаракин / Школьные технологии, 2010, № 4, С. 132 – 135
34. Википедия — свободная энциклопедия [Электронный ресурс]/ режим доступа: https://ru.wikipedia.org/wiki/Squeak

ПРИЛОЖЕНИЕ 1

ДИАГНОСТИЧЕСКАЯ РАБОТА

ЗАДАНИЕ 1

Допишите алгоритм поиска наибольшей из четырёх величин A, B, C и D.

y := a
если *b > y*
    то *y:= b*
все
если _____
    то _____
все
если _____
    то_____
все

ЗАДАНИЕ 2

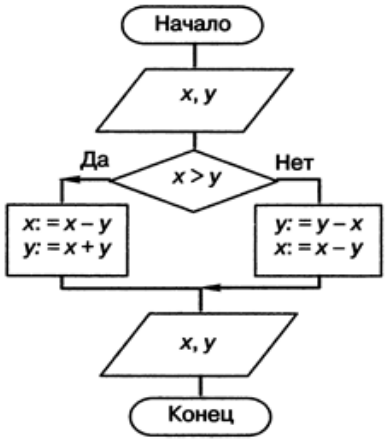
Дан фрагмент линейного алгоритма.

a: =8
*b:=6+3*a*
*a:=b/3*a*

Чему равно значение переменной **a** после его исполнения?
Ответ: a:=_____

ЗАДАНИЕ 3

Исполните алгоритм при ***x = 10 и y = 15.***



Какие значения будут получены в результате его работы?
Ответ: x:=_____, y:= _____

ЗАДАНИЕ 4

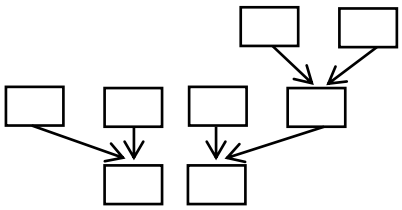
Некий человек должен был перевезти в лодке через реку волка, козу и капусту. В лодке мог поместиться только один человек, а с ним или волк, или коза, или капуста. Но если оставить волка с козой без человека, то волк съест козу; если оставить козу с капустой, то коза съест капусту. Человек все – таки перевез свой груз через реку. Как он это сделал?

Ответ:_____
_____
_____
_____
_____

ЗАДАНИЕ 5

Выберите выражение, для которого подходит данная схема.

Найдите его значение:



A. 299 + 124 : 31 – 132 : 6

B. (299 + 124) : 31 – 132 : 6

C. 299 + 124 : (31 – 132 : 6)

D. (299 + 124) : (31 – 132 : 6)

Ответ:_____

ПРИЛОЖЕНИЕ 2

Оценка уровня сформированности алгоритмических умений обучающихся 8б класса.
Результаты выполнения отдельных заданий.

Умения	**№ задания**	**Учащийся_1**	**Учащийся_2**	**Учащийся_3**	**Учащийся_4**	**Учащийся_5**	**Учащийся_6**	**Учащийся_7**	**Учащийся_8**	**Учащийся_9**	**Учащийся_10**	**Учащийся_11**	**Учащийся_12**	**Учащийся_13**	**Учащийся_14**	**Учащийся_15**	**Учащийся_16**	**Учащийся_17**	**Учащийся_18**	**Учащийся_19**	**Учащийся_20**	**Учащийся_21**	**% выполнения**
1.Упорядочивать действия в алгоритме;	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	0	1	1	86
2.Выполнять линейный алгоритм	2	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	90
3.Выполнять разветвленный алгоритм	3	1	0	1	0	0	1	0	1	1	1	0	0	1	1	1	1	0	1	1	0	0	57
4.Составлять план действий	4	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	0	1	0	1	0	0	67
5.Сопоставить задачу и готовый алгоритм – модель ее решения	5	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	95
Сумма баллов:		***5***	***4***	***4***	***3***	***4***	***5***	***4***	***1***	***5***	***5***	***4***	***4***	***4***	***5***	***5***	***4***	***3***	***4***	***4***	***3***	***3***	
% выполненных заданий		100	80	80	60	80	100	80	20	100	100	80	80	80	100	100	80	60	80	80	60	60	

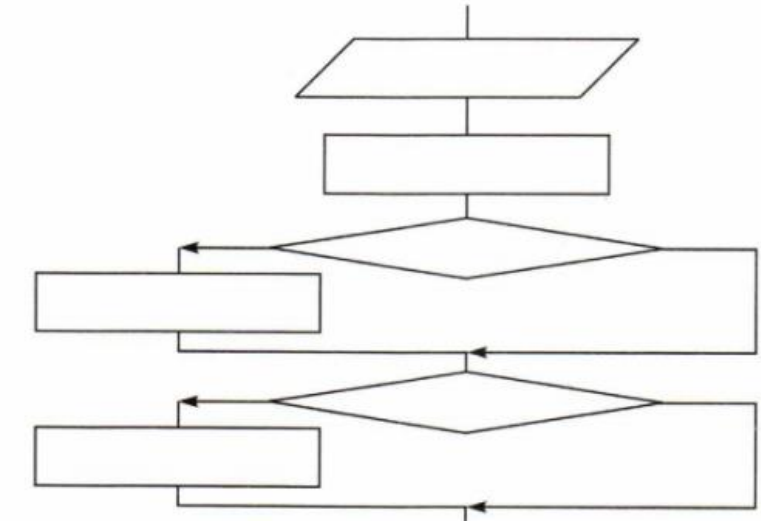
Инструкция: в ходе заполнения карты эксперт выставляет балл по предложенным критериям (0 – задание выполнено неверно, 1 – задание выполнено верно. На основании **шкалы перевода** определяет уровень сформированности алгоритмических умений: **90-100%** – *высокий уровень*; **65-90** – *средний уровень*; менее **65** – *низкий уровень*.

ПРИЛОЖЕНИЕ 3

ДИАГНОСТИЧЕСКАЯ РАБОТА
(вторичный замер)

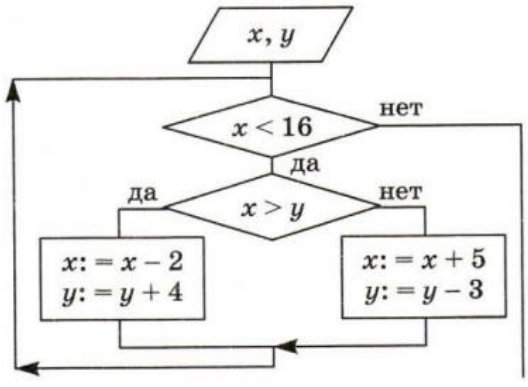
ЗАДАНИЕ 1

Дополните блок-схему, представив в ней алгоритм определения количества четных чисел, имеющихся среди заданных целых чисел a, b и c.



ЗАДАНИЕ 2

Определите значения переменных ***x*** и ***y*** после выполнения алгоритма.



x	y
10	**4**

Ответ:=x_____, y:= _____

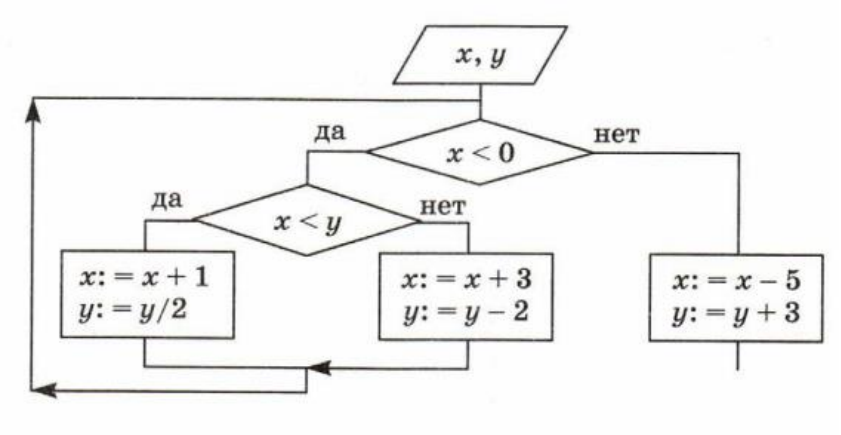
ЗАДАНИЕ 3

Трем неутомимым путешественникам – Андрею, Михаилу и Олегу – надо было переправиться на лодке, выдерживающей массу не более 100 кг, с одного берега реки на противоположный. Андрей знал результат своего недавнего взвешивания – 54 кг и своего друга Олега – 46 кг. Зато Михаил весил около 70 кг. Как им надо было действовать наиболее рациональным образом, чтобы переправиться через реку?

Ответ:_____
_____
_____
_____
_____

ЗАДАНИЕ 4

Определите значения переменных *x* и *y* после выполнения алгоритма.



x	y
9	**5**

Ответ:=x_____, y:= _____

ПРИЛОЖЕНИЕ 4

Оценка уровня сформированности алгоритмических умений обучающихся 8б класса. Результаты выполнения отдельных заданий

Инструкция: в ходе заполнения карты эксперт выставляет балл по предложенным критериям (0 – задание выполнено неверно, 1 – задание выполнено верно. На основании **шкалы перевода** определяет уровень сформированности алгоритмических умений: **90-100%** – *высокий уровень*; **65-90** – *средний уровень*; менее **65** – *низкий уровень*.

Умения	**№ задания**	**Учащийся_1**	**Учащийся_2**	**Учащийся_3**	**Учащийся_4**	**Учащийся_5**	**Учащийся_6**	**Учащийся_7**	**Учащийся_8**	**Учащийся_9**	**Учащийся_10**	**Учащийся_11**	**Учащийся_12**	**Учащийся_13**	**Учащийся_14**	**Учащийся_15**	**Учащийся_16**	**Учащийся_17**	**Учащийся_18**	**Учащийся_19**	**Учащийся_20**	**Учащийся_21**	**% выполнения**	**средний % выполнения**
3.Выполнять разветвленный алгоритм	2	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	86	81
	4	1	0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1	1	1	1	76	
4.Составлять план действий	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	86	83
	3	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	0	1	1	81	
Сумма баллов:		***4***	***3***	***4***	***4***	***1***	***4***	***2***	***2***	***4***	***2***	***4***	***3***	***3***	***4***	***4***	***4***	***3***	***3***	***3***	***4***	***4***		
% выполненных заданий		100	75	100	100	25	100	50	50	100	50	100	75	75	100	100	100	75	75	75	100	100		